Facial Emotion Recognition from Video Data Using Deep Learning and Computer Vision Techniques

University of HUDDERSFIELD

Inspiring tomorrow's professionals

Author:
Dirdh Prafullkumar Patel

MSc. Artificial Intelligence

U2366489

Under the Supervision of

Dr. Tianhua Chen

Abstract

This Report presents the development and implementation of a facial emotion recognition system using deep learning techniques, aimed at accurately detecting and classifying human emotions from video inputs. The research involved a comprehensive process beginning with the collection of diverse datasets, including the VGU , IIMI and FER2013 datasets, which provided a wide range of facial expressions across different demographic groups. To ensure the robustness of the model, data preprocessing steps such as class balancing through weight adjustments and data augmentation were employed, alongside the normalization and resizing of images to 224x224 pixels. Three deep learning models were explored: VGG-16, ResNet-50, and a hybrid model combining features of both. These models were trained using a transfer learning approach, initially on the VGU and IIMI dataset, followed by fine-tuning on the FER2013 dataset to enhance generalization capabilities. The ResNet-50 model demonstrated superior performance, achieving an accuracy of 78% on the FER2013 dataset, surpassing state-of-the-art benchmarks for this dataset. The model's effectiveness was further validated through evaluation metrics such as the classification report and confusion matrix. The final model was deployed via the Gradio Interface , enabling real-time emotion detection in practical applications. This project not only advances the field of facial emotion recognition but also provides a foundation for future enhancements, including the integration of personalized emotion detection systems and the potential combination with facial identification technologies to deliver more individualized and accurate assessments.

Acknowledgements

I would like to express my deepest gratitude to my supervisor, Dr. Tianhua Chen, for his unwavering support, invaluable guidance, and constant availability throughout the course of this project. His encouragement and insightful feedback were instrumental in navigating the complexities of this research and in driving the project to its successful completion. Furthermore, I am also profoundly grateful to my parents for their endless love, patience, and support, which provided me with the strength and motivation to persevere. Finally, I would like to extend my sincere thanks to all individuals who directly or indirectly Helped me for this project. Your assistance, whether through discussions, advice, or moral support, has been greatly appreciated and has played a crucial role in the achievement of this work.

Table of Contents

Table of Figures

1 Introduction

Emotion detection has become a vital and much researched area with its use in human computer interaction and also in mental health assessments and remote communication gauging. With an increase in online communication, especially video meetings, it is now pivotal to discover and recognize emotions from video feeds. It is critical to be able to interpret emotions through video files. This report elaborates on the development of an emotion detection system that is capable to attach video uploads and display full emotional analysis of the video feeds with the necessary graphs.

The system uses modern day deep learning techniques to classify emotions, such as transfer learning with near state-of-the-art accuracy.

One of the many challenges faced with the ever-evolving virtual world due to the omnipresence of Internet-based messaging and communications is the lack of availability of facial expressions, which are nowadays an essential ingredient when engaging in social communications in person. Communicating emotions and feelings through typed text, or even audio-visual sequences, such as phone and videoconferences – imposes limitations that, in time and in certain situations, can lead to stronger misunderstandings and eventually to a reduced level of involvement in all sorts of virtual interactions. For this reason, it is deemed necessary to have a system that can automatically extract emotions from video sequences, especially at the input of online interactions, to provide the user with an estimate of the emotions of their peers and themselves (for online interactions).

The system features the implementation of a method that enables a user to upload a video file and obtain emotion diagnostics based on a thorough analysis. The system finds its application where a therapist who needs to see an event in a patient's life from the inside is not available in real-time, for example when analyzing an event after it happened, or in teaching, frustration analysis or psychological diagnosis. The extracted information from the video enables users to verify their own or other's emotional reactions; to distinguish the reactions from each other; to analyze trends and learn about new patterns. Additionally, the system generates useful reports, sometimes even complete reports with graphs illustrating the trends of the video.

The main aim of the project was to set up a system that could detect emotion from pre-recorded videos and provide a detailed breakdown of emotions deemed present in the video. This system should be quite accurate since deep learning models are being used that have been trained on large datasets.

A number of such design decisions were made to ensure the system is tailored for the task at hand: Examples of these design choices are the strong focus on high-level facial structure features by using pre-trained base models, fine-tuning the initial behaviour of the models through the use of emotion-specific emotion datasets, as well as the fine-tuning of the individual emotion-specific models. In addition to these decisions, the authors made a large number of different choices concerning model design and implementation, with different combinations of architecture and parameters. These choices were first tested in a wide grid of experiments, selecting the models with the best balance of accuracy and computational efficiency.

Beyond the emotion-detection, the system was also designed to report back on the profile of detected emotions throughout the clip, including automatically generated graphs of emotional trends over time (e.g., how often/how strongly are you exhibiting Negative at what time points, etc). From this perspective, the addition of automatically generated graphs for users who might benefit from greater detail (educators reviewing a lecture, or counsellors reviewing a session, for example) can be seen as a significant benefit rather than a source of bias.

Many difficulties had to be overcome, from making the model generally applicable to in different quality videos, to integrating the model into a visualised user-friendly interface that can upload videos and produce more detailed analytical outputs in a short amount of time and ensure usability in a clear way. By pre-processing the data, fine-tuning and integrating carefully into the graphical user interface, these challenges were resolved.

Because emotions can change over time, other vital lessons were learned about the temporal nature of emotion-recognition. Incorporating the analysis of sequences of frames of video as opposed to simply treating each frame as an isolated instance was crucial in differentiating between related emotions such as joy and surprise.

One lesson captured in this project was the need to display the drivers of the analysis visually in intuitive and useful ways to the user. This could be manifested in the form of graphical outputs, such as an emotional trend graph to help see the why and how. A secondary learning outcome from this project was an emphasis on user-centred design, highlighting the importance of a need to strike a balance between technical complexity and practical usability in a real-world setting.

This is followed by the Literature Review which summarises the current state of the art of emotion detection (including studies on typical human approaches as well as some taken with the recent advances in deep learning), followed by the Methodology section which describes the process of collecting the data, the choice of model and the training techniques. It is followed by the Implementation section (technical details of how the emotion detection model has been integrated into a user-friendly interface that allows a user to upload video and create an analysis report) shadowed by the Results and Discussion section (presentation of system performance (emotion detection accuracy as well as the efficacy of the graphical outputs). The next section, Conclusion and Future Work, summarizes the findings and discusses possible avenues of research and development going forward (particularly in improving the models as far as the real-time processing is concerned and increasing the domain applications).

2 Literature Review

Facial emotion recognition (FER) is the technology or method of interpreting emotions from faces: the decoding of emotional states from facial expressions. Recently, the technology base has broadened with the interpretation of emotions from faces by artificial intelligence (AI) (Cai et al., 2020). The assessment of emotional states using face-recognition technology based on AI has recently been demonstrated to work very well (Nidhi et al., 2023). Other recent developments include the assessment of what have been called the 'compound emotion' and 'micro emotions' are combinations of two or more possible states. Micro-expressions are very brief, subtle and unpractised expressions reflecting suppressed feelings or emotions (Cai et al., 2020).

Early perception-from-video and EM methods typically featured handcrafted features and classic machine learning (Calvo et al.,2010). Initial work focused on particular facial features or landmarks (often referred to as key points on the face; Mohanta et al., 2022) and specific emotion responses. For example, the angles between the corners of the mouth or the relative raise of the eyebrows were used to classify happy versus surprise responses. While these techniques were groundbreaking at the time, they are significant limitations, including that they don't generalise well across different populations and are easily confounded by things we do in ordinary, real-world situations, such as changes in lighting or head position, or by the unique way one person's eyebrows might look in comparison with another's.

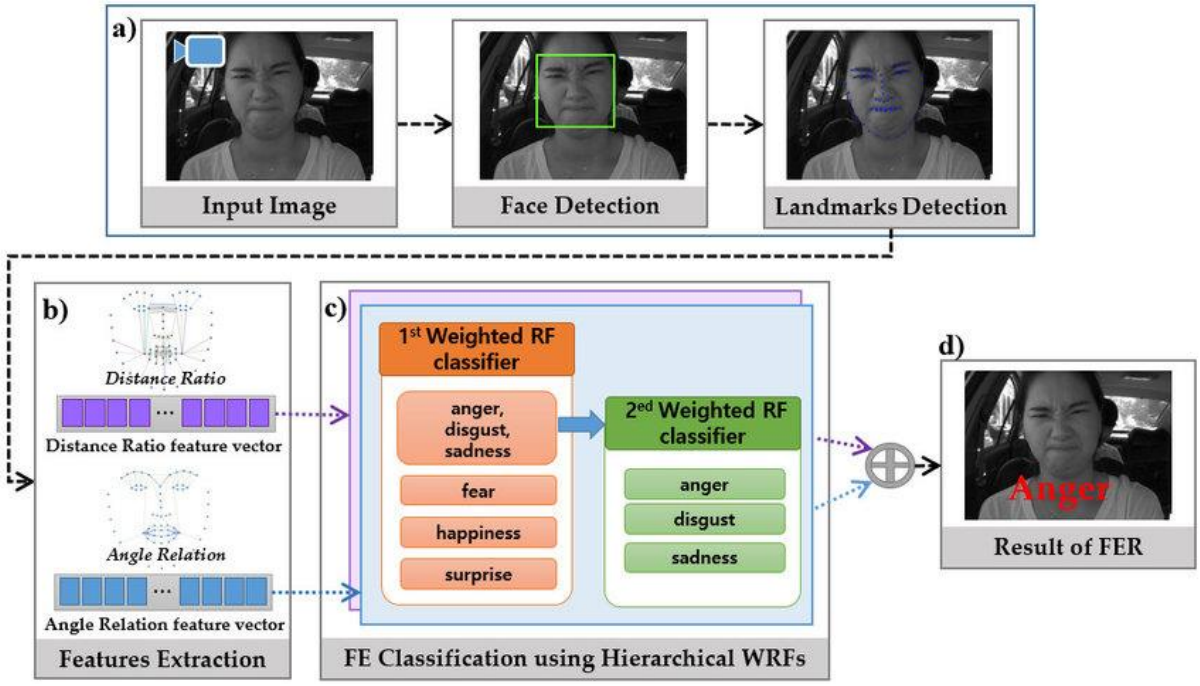


Figure 2-1Example Architecture for Facial Emotion Recognition Using EM Methods(Past (old) Technology)

Unfortunately, these systems also tended to manifest limitations – in particular in real-world situations. If a system had been trained on a dataset dominated by one type of face (for instance, one ethnicity, such as East Asian), it would often not be able to do a very good job with faces of a different type (for instance, African or European) (Mohammad et al.,2022). Nor did it cope well with more complex emotions like mixed or layered emotions, or micro-expressions.

Machine learning, with its earliest versions dating to the 1990s and 2000s, then led to more complex and refined approaches to emotion detection via the analysis of automatic facial features. Algorithms using Support Vector Machines (SVMs) and Random Forests allowed for the lookup of much larger image datasets and a more complex analysis compared to previous approaches (Deshmukh et al., 2017). And since more features equals increased complexity, training these more sophisticated algorithms lead to higher accuracy. But, even these methods required – what is known as feature engineering – the explicit definition of what the emotion detection system should be looking for by the human scientist: they literally picked apart the face to determine what aspects are most relevant to that specific emotion(Kumari et al., 2021).

This changed with the advent of so-called deep learning, most notably the creation of Convolutional Neural Networks (CNNs) in the early 2010s, which made it possible to automatically extract relevant features from raw image data. While deep learning is transforming many fields, it is particularly useful in image recognition tasks because it enables researchers to do what they are not able to do manually: extract subtle, high-level features(Nidhi et al., 2023).

CNNs transformed the situation. Suddenly, it was possible to train systems to detect emotions at unprecedented accuracy levels, even under extreme situational conditions(Li et al., 2017. Systems could handle variations in lighting, pose and idiosyncratic features of faces better than other methods. Since then, CNNs have been applied to real-life situations relating to human-computer interaction, mental health monitoring and market research, among many others (Garcia et al,2017).

Most often, the model used is CNNs, which have been able to achieve good results when applied to several benchmark datasets. For instance, when CNNs were applied to the benchmark dataset of Emotion Recognition, Manalu et al(2024) reported a 80 per cent accuracy rate and an average F1-score of 0.75 when tested on feelings such as happy, sad and anger. They described this as 'outstanding performance'. When compared with traditional methods, using deep learning models such as CNNs makes unnecessary the human task of feature extraction and, more importantly, results in superior overall performance on different benchmark datasets. That, again, is an empirical observation: deep learning is a better way of modelling the intricacies of emotional expression.

One such approach is known as transfer learning, which is a key technology behind breakthroughs in facial emotion detection, which leverages transferred prior knowledge in facial expression in related datasets to overcome small-data bias. Specifically, to this end, we have developed a novel multisource transfer learning method to improve model performance on sparse-annotated data by leveraging knowledge from multisource datasets, learning better multivariate correlations among source tasks, and optimising promotion performance for target tasks (Sarkar et al, 2023). Capturing the group correlation effectively and making few-shot adaptation more resistant to negative transfer. Besides helping in boosting the performance of training models via transfer learning, transfer learning can also be applied toward effective few-shot adaptation. We discovered that few-shot adaptation, using transfer learning, is capable of nearly achieving an elevation in performance compared with the state-of-the-art (SoA) related methods in the literature.

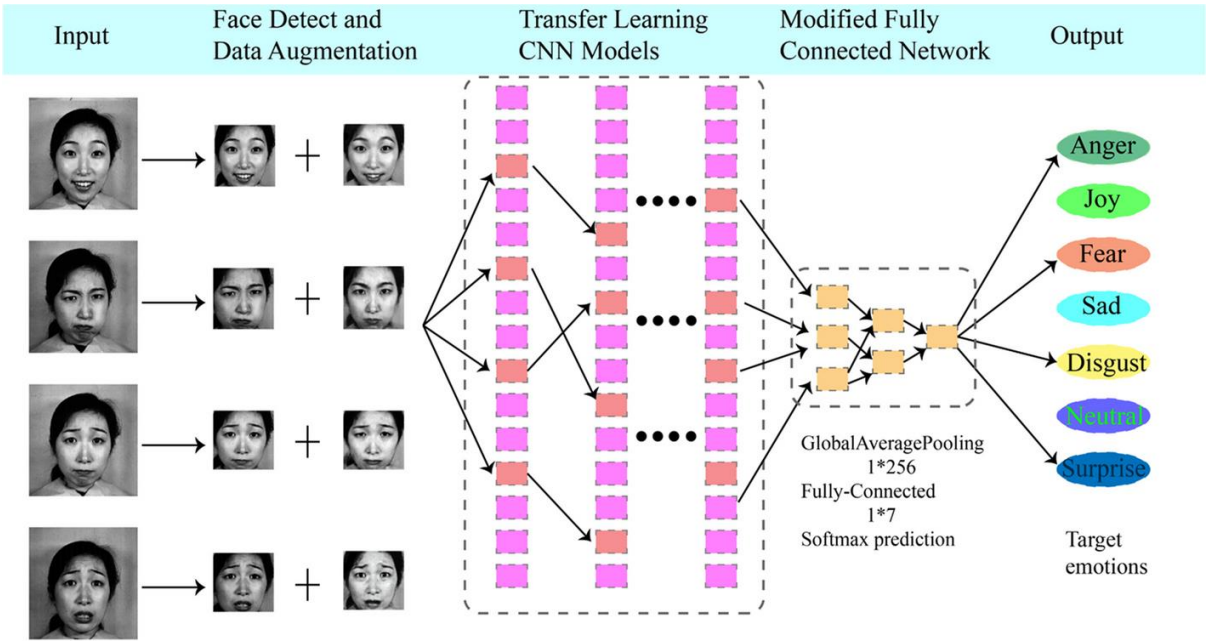


Figure 2-2 Example Architecture for Facial Emotion Recognition Using Transfer Learning method (Current) Gan(2018)

Therefore, transfer learning would be a useful and beneficial for the problem of facial emotion detection as compared to the standard convolutional neural networks. By using the power of pre-learning model along with the data specific to the domain could be an improved performance in the concern problem. An advantage of TL is that one can increase performance under different scenarios where data is scarce. Multiple datasets can be combined to better generalise.

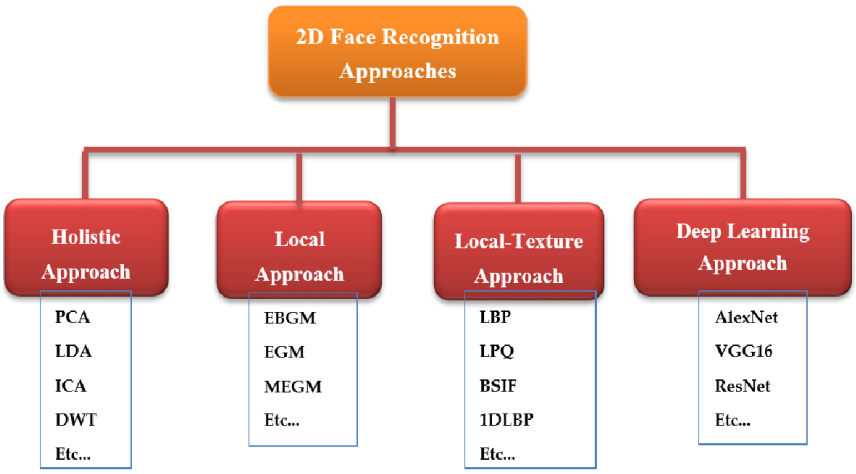


Figure 2-3 Approaches Used for facial emotion Detection(Past and Current)

2.1 Techniques and Algorithms:

VGG16 is a network that in 2014 proved that deeper networks can perform better in image recognition tasks (Simonyan, 2014). VGG16 was the first at par with the ILSVRC, the ImageNet Large Scale Visual Recognition Challenge. This success allowed it to

become the go-to network for various computer vision tasks such as emotion detection. Its architecture shows many similarities with simpler networks, yet it allows for solving complex challenges. For this reason, it became the favourite among both academics and practitioners who wanted to apply machine learning to other tasks than the ILSVRC(Zhou et al., 2024)..

The subsequent leap was ResNet50v2, belonging to the ResNet family of systems first introduced in 2015. This architecture addressed the problem of the vanishing gradient that had limited the depth of neural networks, enabling the creation of much deeper and more powerful models (He et al, 2016). Earlier it was very hard to detect emotions nuanced and complex emotional expressions due to limitations in the training algorithm. An example is the imitation of various emotions by playwright above Subsequently, ResNet50v2 engineered the perfection of emotion detection and allowed us to recognise complex emotions or mixtures of emotions from facial expressions(Saxena et al, 2020).

This progression, from a set of handcrafted features, to the VGG16 and ResNet50v2 deep learning models, also led to increased robustness in the detection of emotions. This means that modern systems can generalise to a wide range of individuals, lighting conditions and cultural contexts (Srinivasan et al, 2018). They can do this because of all the training data that went into the algorithms. Improved robustness means that emotion detection technology can now be useful for some real-world applications, such as improving user experiences on digital devices, or as a tool for psychological assessments (Koole et al, 2009).

While VGG16 and ResNet50v2 both exhibit high success in Image Classification for Facial Images, the networks are both trained differently and have vastly different architecture. VGG16, as the name suggests, has 16 layers with most of them being shallow 3x3 convolution layers that learn to squeeze densely into the features. Because VGG16 is so shallow, it is easy to compute but this simplicity limits the complexity of features that may be learned. By comparison, ResNet50v2 comprises residual connections that enable networks to have significantly deeper structures (50 layers), and encourage the ability to learn more complex features by circumventing the issues of vanishing gradients (Choudhary et al., 2024). The resulting model therefore both converges more quickly, and extracts more accurate features, particularly features involving masked faces from the COVID-19 period.

Transfer learning is essential to have better results for emotion detection. For our case, they have two different trained models as they built with two different levels of complexity: VGG16 and ResNet50v2. Both models were trained in a challenge in ImageNet website, which has millions of images of a wide variety of objects. In this challenge, someone could have an image of a chimpanzee in one might have a baby or an old man. VGG16 model's design avoids overfitting because it was made with smaller datasets but with a reasonable depth. A more recent model was developed and is very different from others in two major ways. First, the model is much deeper than AlexNet or VGG16, so it has more parameters to fit the training data. Second is that ResNet50v2 was designed to help with problems that deep models are at an advantage in training (eg, large and varied datasets), this is very clear while analysing the results that these models achieved in the ImageNet challenge. Shown in Mascarenhas et al.(2021), where see the performance of these models.

2.1.1 Architecture of VGG16:

VGG16, a network created by the Visual Geometry Group at the University of Oxford, is an example of a popular and influential CNN which is intended for use in computer vision (Simonyan, 2014). Although it is an older design, it's relatively simple compared with more recent networks, and highlights features that have been carried forward into subsequent designs.

The VGG16 network contains 16 layers with weights that can be trained: 13 convolutional layers and three fully-connected layers. The amazing thing about VGG16 is that it uses very small 3x3 convolutional filters (filters with 3 rows and 3 columns) at every layer of the network. This was partly motivated by the fact that several very small 3x3 filters could stack together to achieve the same receptive field as one big filter while reducing the number of parameters and increasing the depth(Liu et al., 2017).

At the end of every convolutional layer in VGG16 there's a Rectified Linear Unit (ReLU) activation function. ReLU is a way to make the model non-linear so it can learn more complex patterns. Max-pooling layers help to shrink the spatial dimensions of the feature maps, which down samples the input so it occupies less space while still retaining the most important information (Scherer et al., 2010).

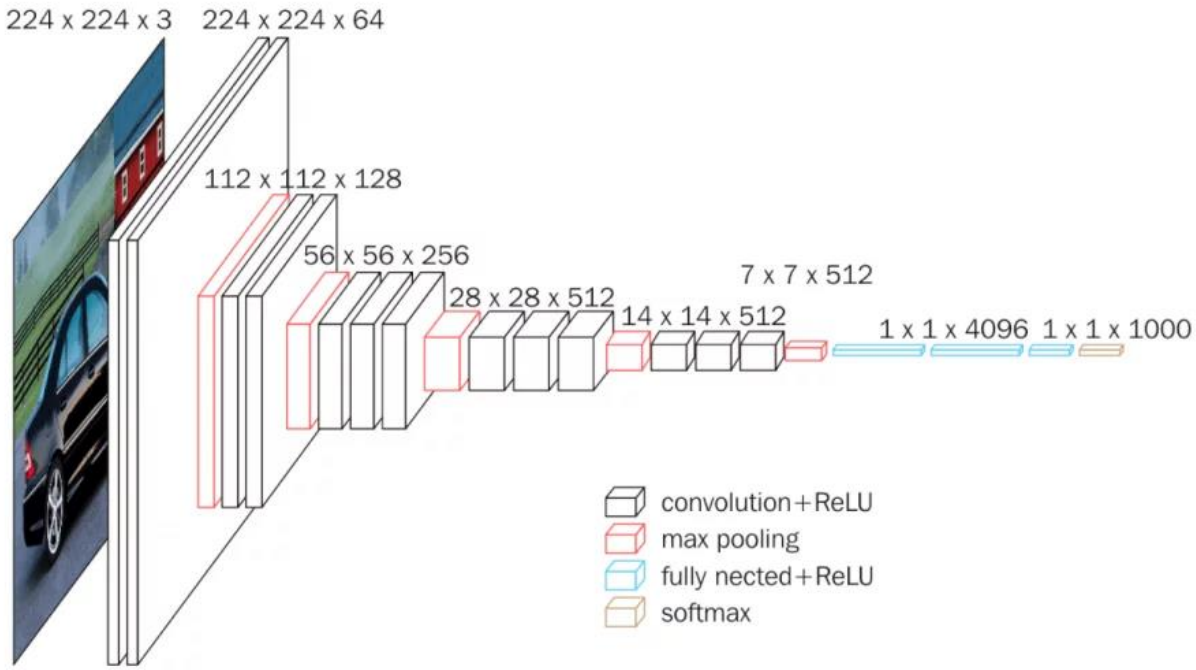


Figure 2-4 Architecture of VGG16

The architecture of VGG16 can be broken down as follows:

1. Input Layer: Accepts 224x224 RGB images.
2. Convolutional Blocks: The network consists of five convolutional blocks, each containing two or three convolutional layers followed by a max-pooling layer.
    - Block 1: Two convolutional layers with 64 filters each, followed by max-pooling.

- Block 2: Two convolutional layers with 128 filters each, followed by max-pooling.
- Block 3: Three convolutional layers with 256 filters each, followed by max-pooling.
- Block 4: Three convolutional layers with 512 filters each, followed by max-pooling.
- Block 5: Three convolutional layers with 512 filters each, followed by max-pooling.

3. Fully Connected Layers: Three fully connected layers follow the convolutional blocks.
    - The first two have 4096 units each.
    - The final layer has 1000 units (for ImageNet classification, though this can be modified for other tasks like emotion detection).
4. Output Layer: Softmax activation for classification.

This uniformity was also one of VGG16's advantages: because 3x3 filters are employed consistently over many layers of the network, both analysis and changes to the architecture become easier (Wang et al, 2021). The uniformity of feature detectors in the early layers of the network makes it easier for them to learn features at multiple levels. For example, the early layers of VGG16 typically learn to detect edges, shapes and coarse textures, while deeper layers combine them to recognise features with greater ecological validity such as the curvature of a smile or the furrow of a brow, which might be more relevant to the task of emotion detection.

2.1.2 Architecture of ResNet50:

ResNet50, from the family of Residual Networks (ResNet) developed by Microsoft Research, is a key step forward in the evolution of deep learning architecture.ResNet50 (He et al, 2016) was applied to address what's known as the problem of Gradient Vanishing , which is the tendency for deeper networks to have higher error in training.

The innovation of ResNet is the establishment of residual learning: rather than learning a mapping from inputs to outputs, ResNet layers are rewritten to learn a residual function with respect to the inputs of the layer (Moravčík et al., 2022). This is done using skip connections (or shortcut connections) that allow the original input to bypass one or more layers.

The architecture of ResNet50 can be broken down as follows:

1. Initial Convolution and Max Pooling:
    - 7x7 convolution with 64 filters and stride 2
    - 3x3 max pooling with stride 2
2. Residual Blocks: The network consists of four stages of residual blocks.
    - Stage 1: 3 residual blocks, each with 3 layers (64, 64, 256 filters)

- Stage 2: 4 residual blocks, each with 3 layers (128, 128, 512 filters)
- Stage 3: 6 residual blocks, each with 3 layers (256, 256, 1024 filters)
- Stage 4: 3 residual blocks, each with 3 layers (512, 512, 2048 filters)

3. Global Average Pooling
4. Fully Connected Layer: 1000 units (for ImageNet classification, modifiable for emotion detection)

Each residual block in ResNet50 follows a "bottleneck" design:

I. A 1x1 convolution to reduce the number of channels

II. A 3x3 convolution

Another 1x1 convolution to restore the number of channels

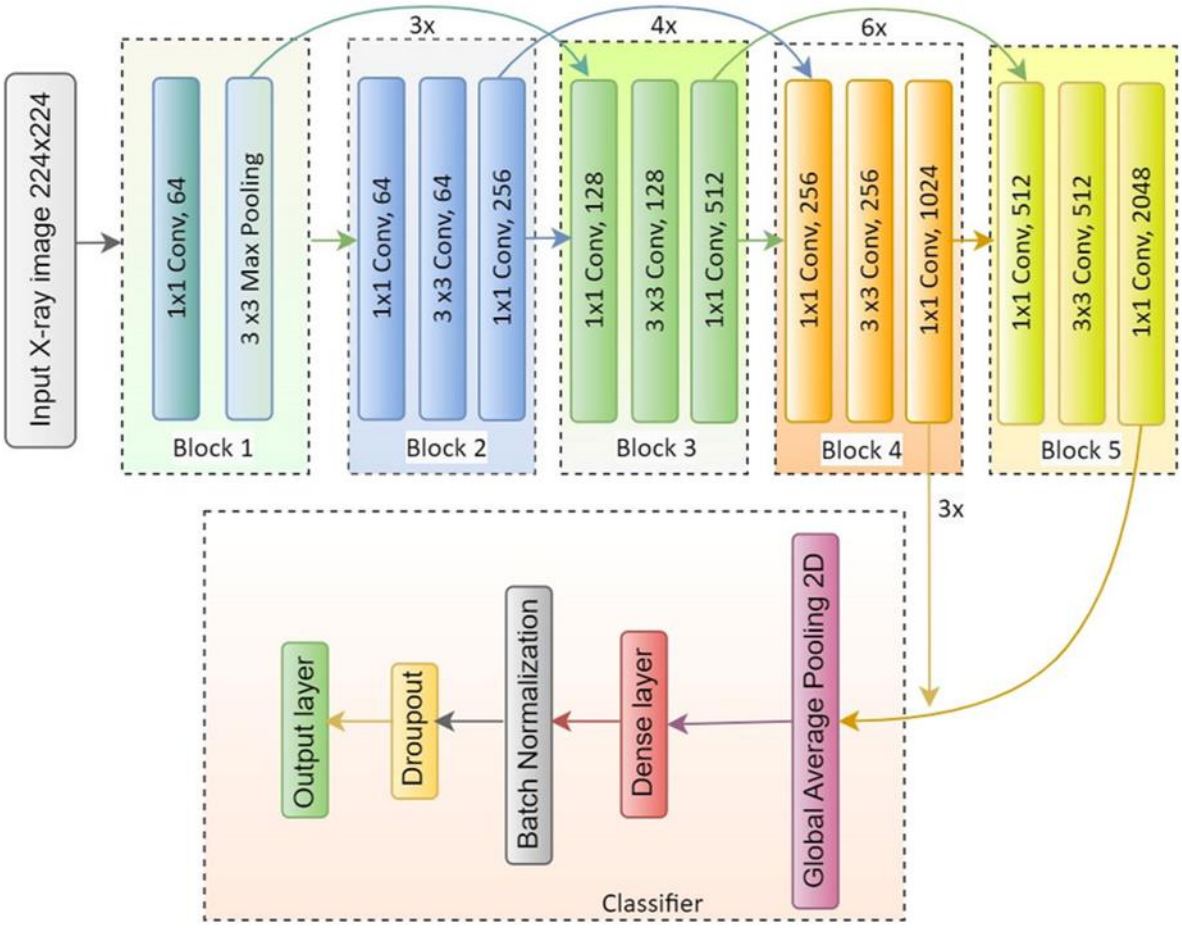


Figure 2-5 Architecture of Resnet50V2

The skip connection adds the input directly to the output of these three convolutions, and then they're all passed through a ReLU activation. In essence, the network easily learns to perform the identity function if required and 'skips' layers(Veit et al., 2016).

The design feature of ResNet50 that makes it able to train very deep networks well is the existence of the skip connections. During backpropagation, these skip connections offer multiple paths for gradient flow. They mitigate the vanishing gradient problem that plagued earlier deep architectures. Now equipped with the ability for its weights to be trained, ResNet50 could learn both low-level features (edges, textures) and high-level, abstract features that are necessary for detecting emotion(Yang et al., 2017).

2.2 Gradient Vanishing Issue for Deep Learning Models:

The problem of vanishing gradients as a fundamental obstacle to training deep neural networks has been studied in relation to network architectures such as VGG16 versus ResNet50 (Zhang et al, 2019). As gradients are backpropagated up the stack, they tend to get very small, which stops the network from learning.

The vanishing gradient problem is one of the most severe impediments to the training of deep neural networks, since propagation of the gradients, that is, the values of the errors, through the network layers back to the inputs to be used to update the weights at the synapses, diminishes exponentially the further back in time we go in the network layers. Vanishing gradient problems in training deep neural networks slow down the learning, or even halts the learning ability of the neural network (it is unable to learn from the data). Finding ways to overcome the reasons for the vanishing gradient problem have improved the technologies of deep learning.

To understand this problem mathematically, let's consider the chain rule used in backpropagation:

$\partial L / \partial W = \partial L / \partial y * \partial y / \partial x \cdot * \partial x / \partial W$

Where:

L is the loss function

W represents the weights of the network

y is the output of a layer

x is the input to a layer

In a deep network, the backwards chain of partial derivatives can get very long. If each partial derivative is small (less than 1) the product gets exponentially smaller the further back we go through the network. If the partial derivatives are large then we have the exploding gradient problem instead(Bengio et al., 1994).

To take a most common example, consider a network with the sigmoid activation function, whose maximum derivative is 0.25. With 100 layers in our network, assuming the worst case of this maximum derivative of 0.25 in every layer would still give a gradient at the first layer of (0.25)^100, which is approximately 1e-60 – that is, zero to any computational precision.

This problem is especially acute in architectures like VGG16, which use a simple stack of layers. As the network gets deeper, gradients can get so small that the weights in the early layers of the network essentially stop updating during training (Glorot & Bengio, 2010). This can lead to poor learning in early layers and can cause the networks to never converge or to converge to suboptimal performance.

2.3 How to Solves the Gradient Vanishing Issue:

Residual learning addresses the vanishing gradient problem directly, by passing shortcuts over one or more layers, directly connecting input to output. This allows gradients being propagated during backpropagation to overcome fewer layers, and therefore the magnitude of the gradient being propagated backward is much closer to the original size (and hence much less likely to vanish entirely and halt any further backpropagation). This way, by preventing the gradient loss that would happen when propagating further, deeper through more layers, models can become vastly, vaster, deeper. But instead of losing their ability to find complex relationships in the data (because of vanishing gradients), they have that ability preserved Alzubaidi et al (2021) .

Mathematically, this can be expressed as:

$y = F(x, \{W_i\}) + x$

Where:

y is the output of the layer

x is the input to the layer

$F(x, \{W_i\})$ represents the residual function to be learned

$\{W_i\}$ are the weights of the layer

The x add-on to the $F(x, \{W_i\})$ forms the critical shortcut, which in residual learning allows passing the gradient (backpropagation) directly through the network, including bypassing potentially faulty layers (Li et al. 2018).

To understand why this helps, let's consider the gradient flow in a residual block:

$\partial L/\partial x = \partial L/\partial y \cdot (\partial F/\partial x + 1)$

The '+1' in this derivative comes from the nucleus of the shortcut connection (x) a binding site of the identity function (x). This NODE+1 makes sure that no matter how small the partial derivative $\partial F/\partial x$ gets, the gradient can flow up the '+1' term in the derivative, bypassing a possible cancellation of the signal.

2.4 Benefits of Residual Learning:

It is Easier to Optimise: if the optimum is closer to an identity mapping than to a zero mapping then it's easier for the network to train to very small perturbations of the identity function(He et al,2016).

Improved Gradient Flow: Shorter pathways through the shortcut connections mean that the gradient is free to backpropagate error signals more quickly (Zhang et al., 2016).

Plasticity in Depth: Residual networks can be made substantially deeper, as they can 'skip' unnecessary layers by learning to output the same thing at the beginning and end (Veit et al, 2016).

2.4.1 Skip Connections:

Skip connections, which are also known by the unfortunate term of 'shortcut connections', are illustrated in ResNet(He et al., 2016), which is a residual network in which the

outputs of identity-transforming asymmetric (ie, there is no mirroring) modules with some number of layers can be passed on directly to later layers further into the network, thus 'skipping over' the intervening layers.

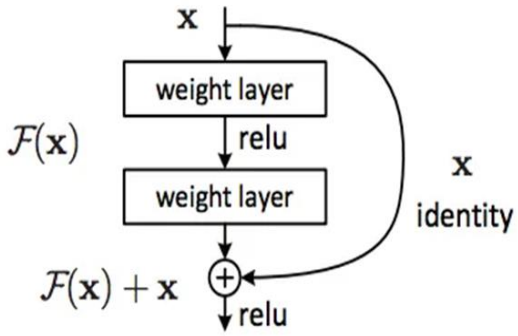


Figure 2-6 Skip Connections

Skip connection's several crucial functions in deep networks:

Gradient Flow: They serve as highways for gradient backpropagation to get around the vanishing gradient problem Gradient will still flow even if F(x, W) gives you something close to zero because all the backpropagations will go through the identity mapping (x) (Oyedotun et al.,2021)

Preservation of information Skip connections help the network to pass forward low-level features that can be lost in VAs with very deep architectures, useful in emotion-detection and other tasks, which require attention to the low-level features (such as edge detection) as well as the high-level features (such as overall facial configuration – see the main text). (Huang et al, 2017)

Ease of Optimisation: They make it easier for the network to learn identity mappings. If the layer turns out not to be helpful, the network can just learn to bypass it by setting F(x, W) ≈ 0 (He et al., 2016).

Better Feature Reuse: Thanks to skip connections, later layers in the network have direct access to features from earlier layers (Huang et al., 2017).

2.5 Benefits of ResNet50v2 over VGG16 for Facial Emotion Recognition:

ResNet50v2, a variant of the original ResNet50, enjoys many advantages over VGG16, especially for the task of emotion detection:

Deeper Architecture: ResNet50v2 has 50 layers, while VGG16 has 'only' 16 layers. This depth enables ResNet50v2 to learn richer hierarchical representations, which is indispensable for discriminating two expressions of closely related emotions from each other. For instance, VGG16 may outperform ResNet50v2 in classifying basic emotions, but the latter can possibly draw a finer distinction between contempt and disgust, which share similar facial features(He et al 2016).

Better Gradient Flow: As a result of the residual connections, the vanishing gradient problem is less pronounced for ResNet50v2. This influences training, allowing for

accurate training of earlier layers of the network, enhancing the learning of useful features from the raw data. In emotion detection, this will generate better use of subtle details in the face. These subtle details can be the most accurate indicators as to the true state of the individual(

Parameter Efficiency: ResNet50v2 (25 million) has fewer parameters than VGG16 (138 million) even though it's structurally much deeper. This could make it more computationally efficient than VGG16, not only because all things being equal deeper nets should be more efficient given the same set of parameters but also since ResNet50v2 drastically reduces the chances of overfitting (discussed below) especially when applied to smaller datasets that are commonly available in emotion recognition tasks (Choudhary et al., 2024).

Better generalisation: The ResNet50v2 architecture helps a facial emotion detection model make better predictions on unseen data. This can be especially useful in emotion detection – a vision-based task that generates better predictions over different people, lighting conditions and even cultures (Bhatia et al, 2022).

Feature Reuse: The skip connections allow ResNet50v2 to re-use features at different stages of the network. This means that low-level features (such as edges, textures and the like) as well as high-level features (such as general facial configurations) can feed directly to the end of the network and help with the final classification, which helps us to detect emotions more robustly (Huang et al., 2017).

2.6 Professional, Legal, Ethical, and Societal Issues in Facial Emotion Detection Systems

Facial emotion action recognition systems – systems that utilise AI to identify and map a human's emotion to a corresponding physical cue based on their facial expression – introduce numerous professional, legal, ethical and societal concerns, which are discussed as follows and accompanied by suggested procedural steps for risk mitigation.

2.6.1 Professional Issues

Facial emotion detection technology has significant implications across various industries, including healthcare, security, and human resources. These developing and deploying these systems must make every effort to ensure their reliability, accuracy and ethical use.

Detecting emotions can be useful for mental health diagnosis in healthcare settings, for providing better patient care, and for monitoring patients remotely. However, if emotion recognition systems make mistakes, the detection error might cause misdiagnosis or inappropriate treatment in healthcare settings (Kaur et al., 2022). For example, if a psychiatrist uses emotional detection to enhance the diagnostic process for depression, they will need to understand the extent to which the system is effective before relying on it. They would also need to make sure that the system is additional to, rather than instead of, conventional diagnostic practices.

In a human-resources context, concerns have been raised about using facial emotion detection for hiring or performance-evaluation decisions (Chamorro-Premuzic et al, 2016). While HR professionals shouldn't rely on emotion-detection tools as decision-making

algorithms, they certainly need to make sure that emotion-detection systems are not discriminating unfairly against job candidates.

2.6.2 Legal Issues

There are significant legal and ethical issues around the use of FEDT technology, including data privacy and consent. Data-protection legislation, such as the General Data Protection Regulation (GDPR) in Europe, stipulates strict rules about processing personal data, including biometric data such as facial displays (Wachter & Mittelstadt, 2019). Organisations are obligated to obtain 'specific, informed and unambiguous' consent from users and inform these users about their right to access and request deletion of their data. For instance, in the EU, companies that employ emotion-detection technologies must still play by these rules to avoid fines (Kaur et al ,2022).

Legal challenges are another concern, such as those arising from abuse of emotion detection for surveillance or unlawful storage of data (Martinez-Martin, 2019). For example, if an officer used an emotion-detection system during an interrogation, that person's right to privacy or right against self-incrimination might be violated when such evidence is used against them in court.

2.6.3 Ethical Issues

There is an obvious ethical dimension to the design and application of FEDS. The first is bias. Such systems will inherit biases in the data on which they are trained, around issues of race, gender and cultural differences (Howard et al, 2021). For example, a system trained primarily on Western faces might fail to recognise the expression of emotion among those in non-Western cultures, leading to treatment of groups unfairly, or miscategorization of emotions.

Another ethical issue is non-consensual use of the technologies. Existing emotion detection systems for public spaces or digital platforms, where users are not given the option to not have their emotional states inferred, encroach on individual autonomy and freedom in terms of control over data (Stark and Hoey, 2021). Implementing such systems without prior notification and informed consent violates ethics in terms of privacy and respect for information.

2.6.4 Social Issues

The mainstreaming of this kind of facial emotion detection technology could have far-reaching effects on society, both in terms of the 'norming' of social interactions as well as in terms of privacy, surveillance, anxiety and behavioural change. If the monitoring of employees' emotions becomes a normed part of work culture, workers could experience stress and potential mental health issues as a result of being continually monitored (Martinez-Martin, 2019).

Meanwhile, there is a social imperative for cultural sensitivity. There is vast variation in both the expression and perception of emotion across different cultures, and systems for facial emotion detection trained on data from a single cultural group could misinterpret emotions in other contexts. (Barrett et al, 2019.) A system deployed globally that does not take these into account risks promoting cultural stereotypes, misunderstandings and even offence.

2.6.5 Mitigation Strategies

It is important that these considerations be addressed at the outset to set the course for the future, as reflected in the above grounds of action for responsible application of the technology of facial emotion detection. At the professional level, users and developers would do well to develop trustworthy systems with clear ethical protocols concerning transparency, fairness and user consent(Andreotta et al, 2019). Within a legal framework, systems would need to comply with the emerging global standards for data protection exemplified by GDPR to avoid data misappropriation or any form of misuse that would encroach on the users' freedoms and privacy. Ethically, models should be trained on datasets that are representative and less biased, to enable the development of systems that offer fair and balanced treatment to all users (Ntoutsi et al., 2020). Socially, systems should incorporate cultural sensitivities, while organisations should enhance public awareness of the limitations and implications of the technology.

This means that, by taking these professional, legal, ethical and social issues seriously, developers can strive towards developing useful facial emotion detection systems that are respectful of individual rights and cultural diversity. Through an ongoing dialogue involving ethicists, policymakers and user groups, developers can make sure that these systems conform to the values of the society in which they are implemented and function within acceptable ethical and legal frameworks.

3 Methodology

The overview of this emotion detection project is outlined in the form of an algorithm that follows a structured way of building an emotion classification system. The project begins with the task of Data Collection, where collected the Vietnamese-German University Emotion Recognition Dataset and IIMI Emotional Face Dataset. These datasets contain many classes of different emotions. The next step is Data Preprocessing, where performed Face Detection, Frames Extraction, and Data Argumentation to obtain images with a high quality, resolution, and the same format for training the models. Next, proceed to the stage of Model Selection and Experimentation, where trained and experimented with pre-trained CNN models. The models include VGG16 and ResNet50V2 with transfer learning and tested the results of different unfreezing strategies and fully connected layers for the two networks. Then, trained a Hybrid system by combining the two best models to get more power for emotion detection. The next step is Model Training, where trained the models using VGG16 and ResNet50V2 on the Vietnamese-German University and IIMI Emotion Recognition Dataset respectively and retrained the models on the FER dataset for model enhancement. Training on the FER dataset is the most important phase to upgrade the performance of the models for emotion detection. It was attempted with advanced techniques such as ModelCheckpoint, EarlyStopping, and ReduceLROnPlateau to prevent the problem of overfitting. And also used Evaluation to test the Models' performance after training with VGG16 and ResNet50V2. Using the confusion matrix and classification report classified the emotion into Negative, Positive, and Neutral to show the accuracy and performance. The final step is Deployment, where use the Gradio for building an interactive system where any user can upload the video to analyse the emotion. The result of deployment displayed the emotion information from the videos, such as Line Graph, Bar Graph, and Pie Chart for emotion trends. It also had a CSV file for exportation. The Flow Flow Chartis shown below to illustrate the methodology of emotion detection.

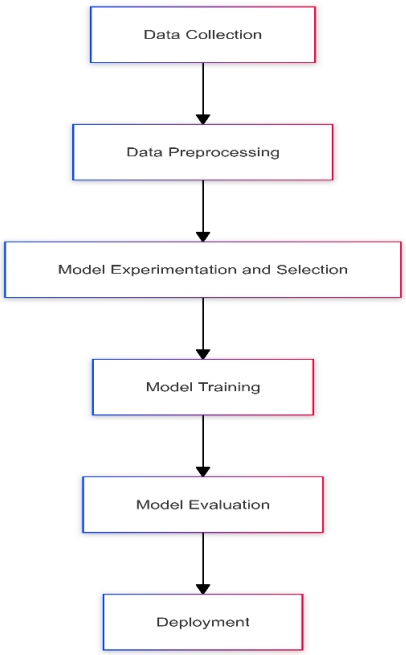


Figure 3-1 Overview Flow Chart of Methodology

3.1 Data Collection and Standardization

3.1.1 Overview of Data Collection

The three datasets adopted to develop the emotion recognition model were: the Vietnamese-German University Emotion Recognition Dataset, the IIMI Emotional Face Database, and the FER-2013 dataset. Such diverse datasets were adopted to cover cultural representation, as well as emotions, which have a large inter-cultural variability. Since each dataset had a discrete structure, a deal of preprocessing and standardisation was required to provide a uniform setup for the model training.

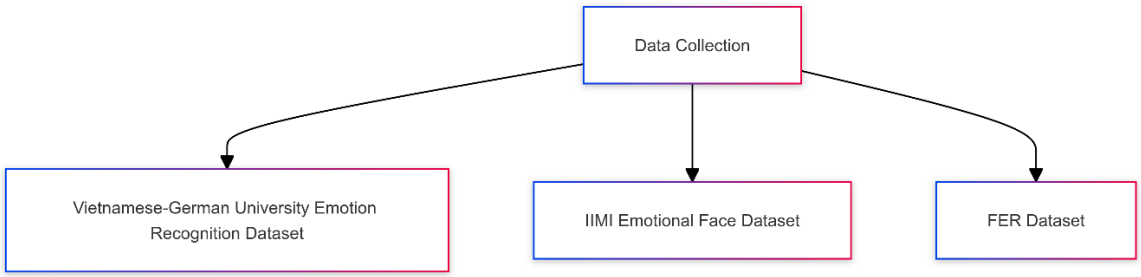


Figure 3-2 Overview of Data Collection

3.1.2 Vietnamese-German University Emotion Recognition Dataset

3.1.2.1 Dataset Description:

The Vietnamese-German University Emotion Recognition Dataset (2022) has 9,892 RGB images, each resized to 640x640 pixels. The dataset consists of eight categories of feelings: Happy, Sad, Contempt, Neutral, Anger, Fear, Disgust, and Surprise. The aforementioned dataset is divided into 8,893 images (90%) and 999 images (10%) for training and testing, respectively (Vietnamese-German University,2022).

Figure 3-3 Sample Images Of VGU dataset

3.1.2.2 Challenges in Dataset Structure:

The first step in exploring this dataset was to unpackage the supplied files. The source provides a CSV file containing filenames and 8 class one-hot encoded labels. Unfortunately, despite that, the provided base directory is not a collection of class-specific directories, which is been considered the standard directory structure for many decades in the branch of machine learning.

3.1.2.3 Data Standardization Process:

Exclusion of the Contempt Class: Since this project didn't have any use for the 'Contempt' class of 1,250 images, disregarded the 'Contempt' class and used seven core emotional classes: Happy, Sad, Neutral, Anger, Fear, Disgust, and Surprise.

Reorganization into Class-Specific Folders: The dataset was standardised by putting all the images into a directory tree with a name 'train' to hold the training dataset and a name 'test' to hold the test-dataset, and each time divided the directories into seven emotional classes according to the labels of training and testing pictures. In other words, sorting the images into seven sub folders, named 'anger', 'disgust', 'fear', 'happiness', 'sadness', 'surprise' and 'neutral' according to their emotional labels and putting them in each folder. According to a practice of computer programming, it was far easier to process images in their respective folders.

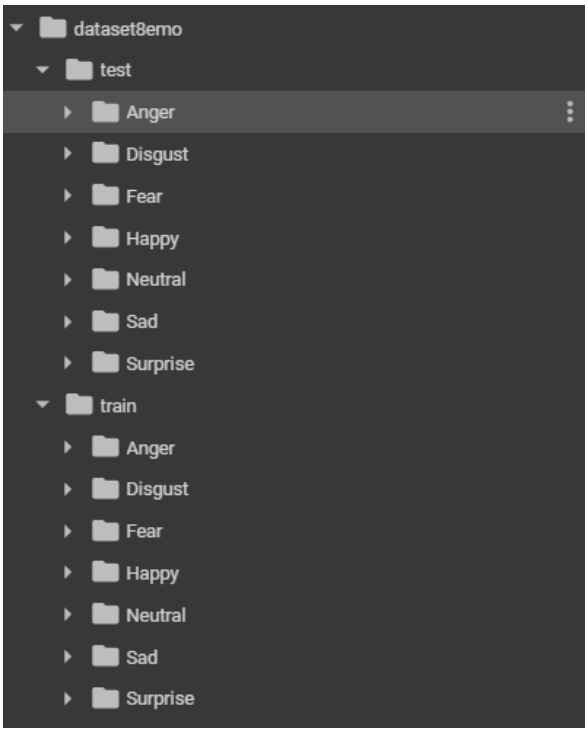


Figure 3-4 Standard Form of Dataset for Image Classification

File Preservation: In particular, the original filenames have been retained and not modified, so at the data output level each row has been associated to a specific image upon input.

3.1.3 IIMI Emotional Face Dataset

3.1.3.1 Dataset Description:

As the name suggests, the IIMI Emotional Face Database (2023) contains 1,302 images of Indian faces, one of seven emotions per image: Anger, Disgust, Fear, Happy, Sad, Surprise, and Neutral.; The images are captured in high resolution which make them further useful for training: the expressions are quite distinct (Shruti Tewari et al,2023).

Figure 3-5 Sample Images Of IIMI dataset

3.1.3.2 Challenges in Dataset Structure:

The original data from the IIMI dataset, for instance, encoded information about metadata such as the code for a participant, their gender and the emotion they might be experiencing (for example: DST_002_F_ANG.jpg). This is useful as a way to keep track of the data but doesn't have the directory-based structure necessary for standard machine learning tasks.

3.1.3.3 Data Standardization Process:

Reorganization into Class-Specific Folders: They were placed in a folder called 'train', which was then subdivided into subfolders dictated by the seven emotional classes. Each image was moved to the appropriate subfolder based on the emotional class indicated by the image name – which followed the standardised format used in model training.

Training-Only Use: The images are so detailed that we decided to use only the IIMI dataset for training; that way, we maximised the available data on the most highly resolved images, which is essential for training models that distinguish emotions accurately.

3.1.4 FER-2013 Dataset

3.1.4.1 Dataset Description:

FER-2013 is a well-known benchmark dataset of 48×48 greyscale images depicting people's faces, with seven emotion labels: Angry, Disgust, Fear, Happy, Sad, Surprise, and

Neutral. The dataset emerged from a Kaggle competition, and it remains a benchmark for emotion-recognition models.

Figure 3-6 Sample Images Of Fer2013 Dataset

3.1.4.2 Challenges in Dataset Structure:

The FER-2013 dataset, in contrast to the others, was provided in the format of a CSV file, with each row representing a pixel of an image, accompanied by its emotion label. These images couldn't be downloaded in the usual image formats but had to be reconstructed from their pixel data.

3.1.4.3 Data Standardization Process:

Folder Organization: Reconstructed face images were deposited into 'train', 'validation' and 'test' folders with those seven categories as subfolders. This standardisation helps training and validation data to be managed easily.

File Preservation: The images were saved with filenames corresponding to their original indices in the CSV file, ensuring traceability, and the filenames stayed consistent with the names of the original dataset.

3.2 Data Preprocessing

Data preprocessing is the first essential step of any machine learning algorithm pipeline: it has a direct impact on the final performance, robustness, and generalisation of a trained model. For a project on emotion detection from emotions in face images, data preprocessing would include several key steps such as data balancing, normalisation and scaling, and the splitting of the datasets into training, validation, and testing sets. Moreover, different data augmentation steps are used to make the training data more diverse and also to avoid overfitting and class imbalance problems. This section describes each of the preprocessing steps in detail.

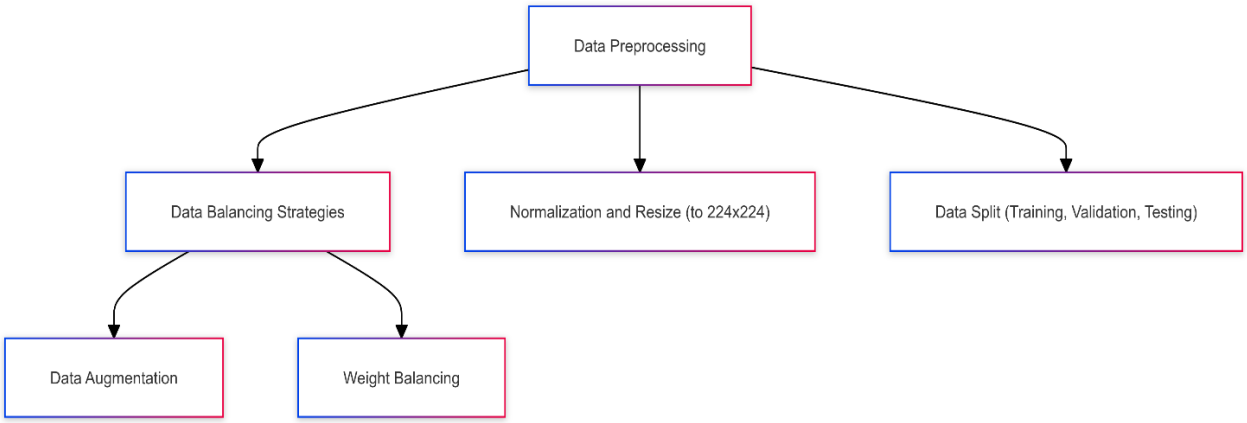


Figure 3-7 Overview of Image Preprocessing

3.2.1 Data Balancing

Another major problem that which is common in any emotion detection model – is that the data can be imbalanced. That means that some emotion classes were far less prevalent, or rarer. A model trained on this data would probably be biased towards the high-frequency classes and perform poorly on the low-frequency classes. Clearly that was a problem for fer-2013 data and needed to address it. If not, the model wouldn't be able to pick up on any emotion, not just the less common ones.

3.2.1.1 Class Distribution Analysis

The first step was to analyse the class distribution of each dataset in detail. It was found that the Vietnamese-German University and Indian Community datasets had a fairly balanced distribution of images across the emotion classes. These two datasets were ideal for training because they required only minor adjustments to achieve balance.

However, this FER-2013 dataset also had a problem with class imbalance: the distribution of emotion labels was very skewed, with, for instance, emotions such as happiness being much more abundant while emotions such as disgust or fear being very underrepresented. This imbalance would have been a source of risk of the model learning to be biased towards the more common emotions, leading to a poor generalisation capacity when it came to emotions that were not present as enough examples.

3.2.1.2 Class-Weighted Approach

a class-weighted approach to counterbalance the skewness in the distribution of instances across the classes in the FER-2013 dataset. In this technique, different weights are assigned to each class based on their relative frequencies in the dataset. The weights are calculated using the inverse of the class frequency so that underrepresented classes get higher weights. The main use of these weights during the training phase is to penalise the loss function if a model misclassifies them. Due to the higher weights assigned to the minority classes, the model gets penalised more severely for misclassifying these classes. This encourages paying closer attention to the less frequent emotions, thereby improving its overall performance on all classes.

The class-weighted approach proved especially helpful in preventing the model from just learning to predict the majority classes, to the detriment of the less represented classes, thus enhancing the model's capacity for learning by balancing its knowledge.

3.2.1.3 Data Augmentation

Besides class-weighted training, data augmentation was employed as an additional technique to counteract class imbalance. Data augmentation consists in generating new training samples by randomly applying transformations on existing images. This has the dual effect of increasing the dataset size and injecting noise into the training samples, which can help the model generalise to new data.

In this work, a few data amplification methodologies like horizontal flip, zoom, rotation, and shift were adopted to simulate various facial expressions that might look realistic, and to ensure the facade data was still aligned with the real-life scenarios.

Mirroring or Horizontal Flipping: This transformation reflects the image about its vertical axis. Effectively, this is a mirror image of the face. Facial recognition is also often trained on mirrored examples of the face. Since the test data will also be mirrored, this increases the size of the training data and helps the model generalise to the different orientation of the face.

Zooming: Each time through the database, there is a small random operation to zoom the recording in or out as if one had suddenly moved closer or further away from the subject's face. This will make the model handle slight differences in the positioning of the face in the frame better.

Rotation: Small random rotations (±15 degrees) were added to account for head tilts and other behavioural variations in pose. The augmentation technique guarantees that the model

3.2.2 Normalization and Data Resize

Normalisation is an important preprocessing step – all input data is scaled to a consistent range, so the model learns better by eliminating irrelevant distinctions in the numbers. In the case of images, normalisation typically consists of normalising pixel values to a range of [0, 1]. Rescaling helps ensuring that no particular feature (or pixel) dominates the training process by virtue of its scale, leading to vanishing or exploding gradients.

3.2.2.1 Pixel Value Normalization:

In this particular project, all the images were normalised by rescaling each pixel value from the original range of [0, 255] to the range of [0, 1] by dividing it by 255. Normalising the pixel values allows the model to receive input data on a consistent scale which is necessary for the stability of the training process, and for the model to converge.

Normalisation can also help to neutralise differences in the lighting conditions and quality between the pictures in the datasets. By bringing all the pixel values into a common scale, the model won't be biased by any difference in illumination or contrast and will learn based on the intrinsic properties of the images.

3.2.2.2 Normalization Across Datasets:

Since the datasets were collected over different time periods, under different illuminations, and by different people, it was crucial to normalise the raw inputs across all datasets in order to minimise the possibility that the model would learn to behave in a way that favours one dataset over another. Normalising the data in the same way reduces the chance that the model will pick up on a dataset-specific artefact, allowing it to focus on learning relevant shape features for emotions.

3.2.3 Splitting Data into Training, Validation, and Testing Sets

Data splitting is vital because it ensures that the machine learning model is trained, validated, and tested on different subsets of the data - a structured approach that helps to assess how well the model is actually performing and how well it will generalise to unseen data.

Data split for this project took place in 2 different stages, which I've outlined below.

3.2.3.1 Initial Training Phase with Vietnamese-German University and IIMI Emotional Face Datasets

Vietnamese-German University Dataset:

- Total Images: 9,892
- Training Set: 8,893 images (90%)
- Testing Set: 999 images (10%)
- To optimize the training process, the 8,893 images designated for training were further divided:
- Training Subset: 7,114 images (80% of the training set)
- Validation Subset: 1,779 images (20% of the training set)

IIMI Emotional Face Database:

- Total Images: 1,302

While the IIMI Emotional Face Database was used completely for training, due to the high resolution and clear images, benefiting the training phase, the images of this database were merged with the training subset of the Vietnamese-German University dataset.

Merged Training Data:

- Training: 7,114 images (From Vietnamese-German University) + 1,302 images (From IIMI Emotional Face Database) = 8,416 images.
- Validation: 1,779 images (from Vietnamese-German University)

This merged data set formed the basis of the first training run to learn a multiplicity of facial expressions and cultural representations.

Testing:

- Testing: 999 images (from Vietnamese-German University)

Retraining Phase with FER-2013 Dataset

After that training period, the model was re-trained with FER-2013, the more closely matched dataset, which has been specifically designed for fine-tuning scaled networks like the one used.

FER-2013 Dataset Structure:

- Training Set: 22,967 images (80% of 28,709)
- Validation Set: 5,742 images (20% of 28,709)
- Final Testing Set: 7,178 images (consisting of the public test set and the final test set)

Retraining Process:

Now, the model was retrained on the 22,967 instance-level images from the FER-2013 training set, which served to deepen its learning, especially by exposing it to large and diverse sets of faces expressing a wide range of emotions.

Validation: The 5,742 images from the FER-2013 validation set were used as part of the input during the retraining to check for the performance of the model and avoid overfitting. This data set helped in keeping the model robust so as to continue learning from the unseen data.

Final Testing: The final results of the model are tested using the test set containing a total of 7,178 images of FER-2013 which is obtained from the training set. It is important to mention that this test set is not used for training or validation process, and it is just used to estimate the performance of real-time situations in terms of emotion detection.

3.3 Model Selection and Experimentation

When it concerns emotion detection using deep learning, the selection and experimentation of models are of utmost importance for having high performance. In this section, describe in detail the architectures and parameters of three types of CNN models: VGG16, ResNet50V2 and a hybrid model, which are chosen and experimented for the study. Through analysing their architectural characteristics and parameter settings, the influences of these parameters on the performance of emotion detection systems will be addressed.

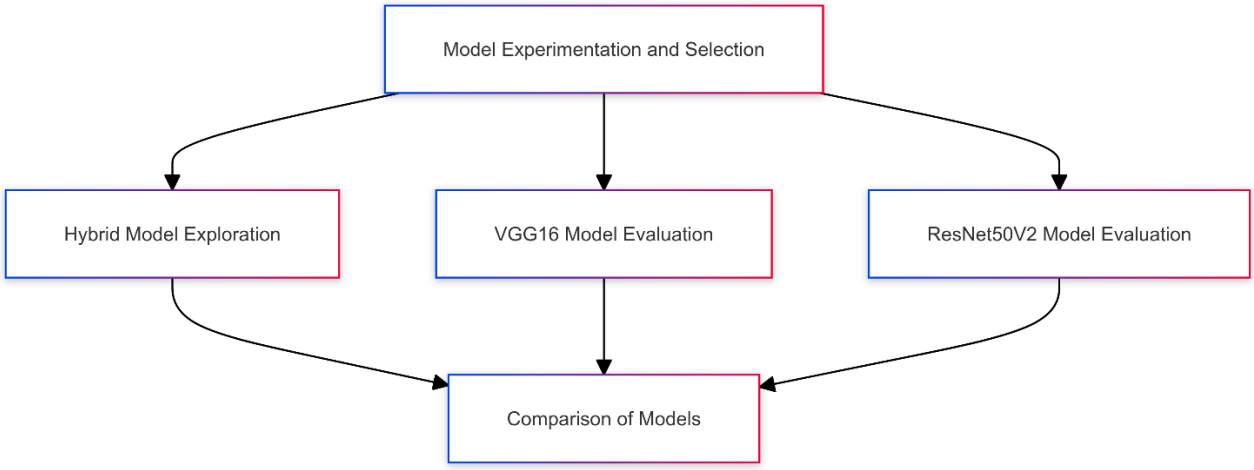


Figure 3-8 Flow Chart of Model Selection Experiment

3.3.1 VGG16 Model

3.3.1.1 Architecture Overview

One of the most widely used vision architectures is called the VGG16 (because it has 16 layers) inspired by the Visual Geometry Group (VGG) at Oxford University, It's such an elegant and simple network. It's easy to understand because it solves the design problem of vision in a very direct way.

3.3.1.2 Detailed Layer Analysis

Input Layer

224x224x3 Image of size 224 x 224. The dimension 224 is the size of the input image, and 3 is the number of channels in an RGB image. 224 x 224 x 3 is a common size of an input for many CNNs and presents the input data in an efficient format that can be processed by the neural network designs in subsequent layers.

Convolutional Base

The convolutional base produces a 7 ×7 ×512 feature. This high dimensional representation of the input images contains the features from simple textures to patterns.

Flatten Layer:

The flatten layer flattens 3D feature map into 25,088, a 1D vector. Dense layer: This layer has 1,000 nodes, in order to let model read the 25,088 parameter.

Dropout layer:

This layer randomly discard 25 per cent of neurons during the training phase to prevent the model from memorising the task.

Dense Layers:

- Highly Dense Layer 1: 256 neurons in this layer take the flattened features, and find higher abstractions on top.
- Dense Layer 2: The subsequent layer with 128 neurons further refines the features.
- Dense Layer 3: a 64-neuron layer serving as an interface to the classification unit.
- Output Layer: The last layer containing 7 neurons corresponds to the emotional categories as predictions for the model.
- Batch Normalization: applied after every dense layer, this normalises outputs from each training batch to stabilise the training process and speed up any computational process.

3.3.1.3 Parameter Details

Total Parameters:

21,180,871 number of parameters throughout the entire VGG16 model, from the convolutional layers to the fully connected layers. The simplicity and depth of the structure of VGG16 significantly increase the number of parameters, which allows the model to capture detailed features and patterns within images.

Trainable Parameters:

Out of 13,552,367 total parameters, 13,544,711 are trainable. With each iteration of the training process, trainable parameters are modified with the goal of learning something useful from the data. Trainable parameters can be found in the convolutional and fully connected layers.

Non-Trainable Parameters:

VGG16 has 7,636,160 non-trainable parameters that correspond to the pre-trained weights in transfer learning. During the training of the new task, these parameters

stay fixed so that we don't tamper with the general features it has learnt for the ImageNet dataset.

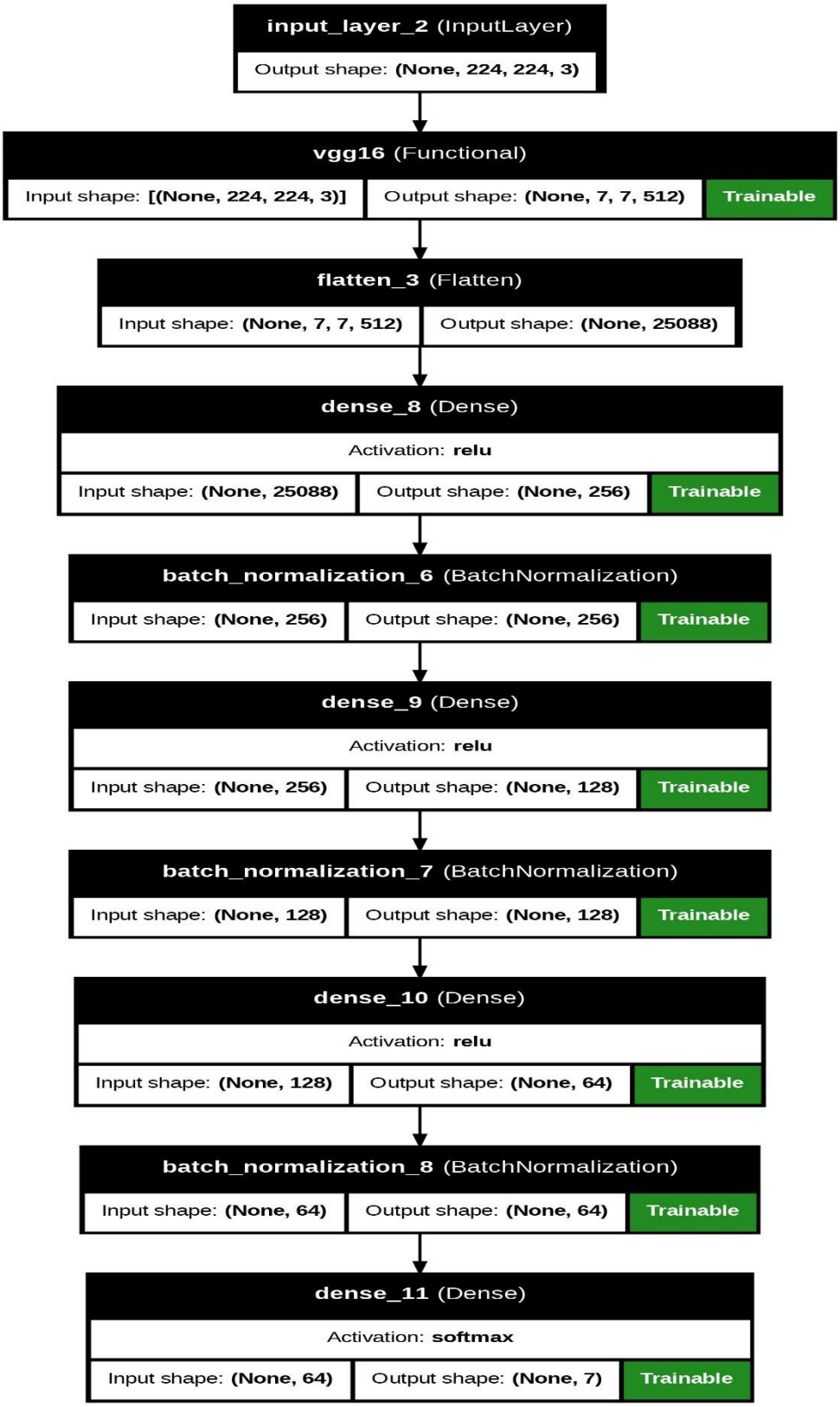


Figure 3-9 Architecture of Vgg16 model transfer learning

3.3.2 ResNet50V2 Model

3.3.2.1 Architecture Overview

ResNet50V2 is a follow-up on the original ResNet architecture that pioneered residual learning using shortcut connections to address the vanishing gradient problem. This model has 50 levels in its depth and capture much more sophisticated features than its shallow counterparts.

3.3.2.2 Detailed Layer Analysis

Input Layer

Images of 224x224x3 are fed into the module, so that the input perceives the sequence with the same spatial characteristics as input into VGG16.

Residual Blocks:

The feature maps of the residual blocks have size 7x7x2048. These deeper features capture a broader distribution of features of the input images.

Flatten Layer

Layer that flattens the high-dimensional feature map into a one-dimensional vector of length 1,00,352 and is passed to the next fully connected layer in the network.

Dense Layers

- Dense layer 1: Extracts additional features from the flattened features from layer 4, using 256 neurons.
- Dense Layer 2: The layer with 128 neurons further processes the features.
- Eventually, the data wends its way into the highest layer (dense 3), where the most neurons 64 are trying to make up their minds as to what the image could possibly be. The final dense 3 layer is the last layer that prepares the data for its classification.
- Output Layer: The final layer with 7 neurons gives the predicted probabilities for the animal's emotion category.
- Batch Normalization: Batch normalization is used after each dense layer in order to make the training more stable and converge.

3.3.2.3 Parameter Details

Total Parameters:

ResNet50V2 has a total of 49,298,567 parameters. Compared to VGG16, this large number of parameters reflects the increased depth of the model (which also has more filters at each stage), which allows it to detect richer and more intricate features from the image.

Trainable Parameters:

42,085,127 This number of trainable parameters indicates that having 42 million parameters is optimal for the ResNet50V2 network when training it to perform the emotion detection task, compared with other available CNN models.

Non-trainable Parameters Non-trainable Parameters 7,213,440 (With pre-trained weights from ImageNet being able to learn emotions without updating them.)

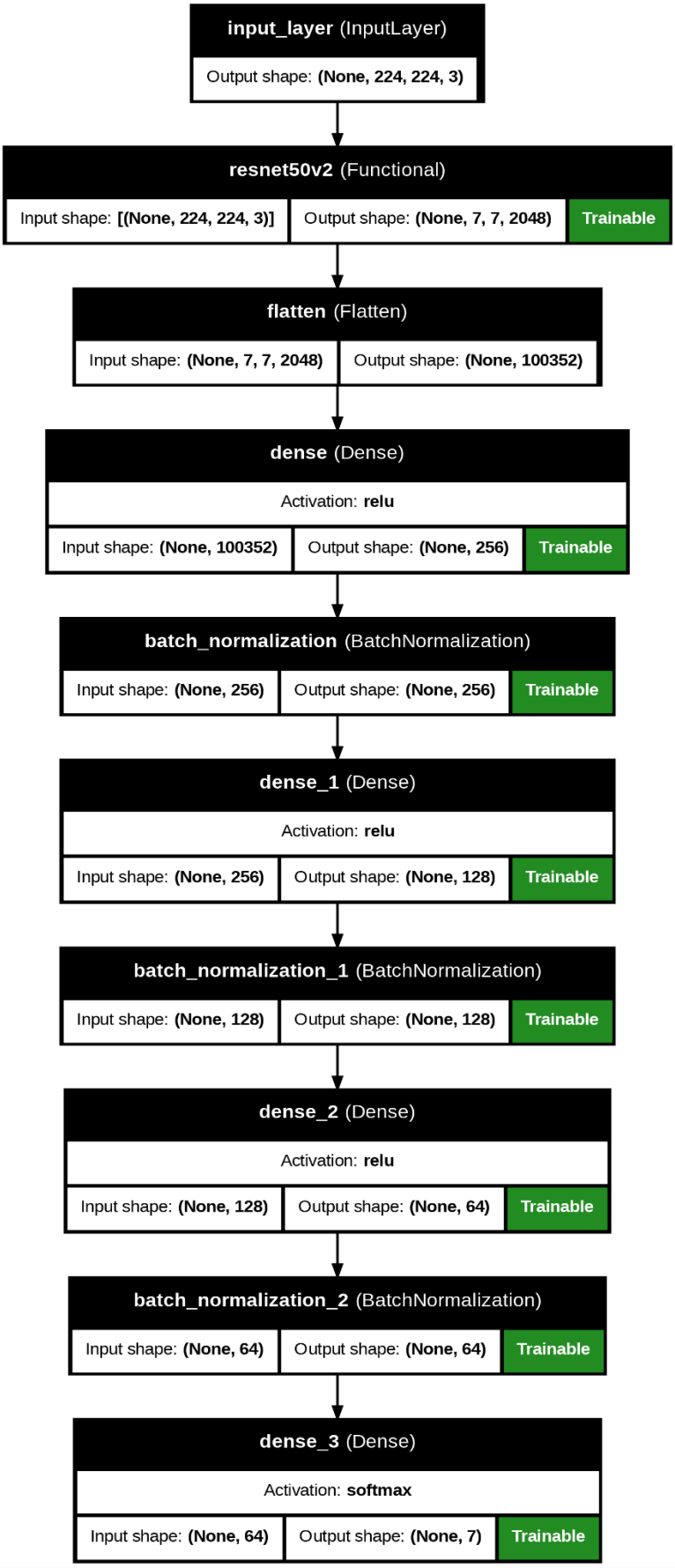


Figure 3-10 Architecture of Resnet50v2 model transfer learning

3.3.3 Hybrid Model

3.3.3.1 Architecture Overview

In the hybrid model, the feature extraction parts from both the VGG16 and ResNet50V2 now work together instead of working separately. This is because they are now combined into hybrid model, and we hope to use the accuracy of both networks.

3.3.3.2 Detailed Layer Analysis

Input Layer:

The input layer takes in images that are 224x224x3 pixels. We chose this size because it is the same as the other models.

Feature Extraction:

- VGG16 Feature Maps: Provides 7x7x512 feature maps.
- ResNet50V2 Feature Maps: Provides 7x7x2048 feature maps.
- Concatenation Layer: Combines these feature maps into a single vector of length 125,440.

flatten layer:

The combined feature map will then be flattened into a one-dimensional vector with 1,25,400 lengths.

Dense Layers:

- Dense Layer 1: With 256 neurons, this layer processes the concatenated features.
- Dense Layer 2: The layer with 128 neurons further refines the features.
- Dense Layer 3: The layer with 64 neurons prepares the data for classification.
- Output Layer: The final layer with 7 neurons provides the emotion classification.
- Batch Normalisation: Batch normalisation is applied after each dense layer to stabilise and speed up the training process.

3.3.3.3 Parameter Details

Total Parameters: This is the most complex model, with a total of 70,435,783 parameters, as it combines the feature maps of both VGG16 and ResNet50V2, which results in a richer feature representation.

Trainable Parameters: 55,587,079 Due to the enormous number of trainable parameters of the hybrid model (55,587,079), it can capture highly detailed features from the data, but it also very easily overfits due to its complexity.

Non trainable parameters: 14,848,704 - that are the non-trainable parameters that include the weights of both - VGG16 and ResNet50V2 - pre-trained on ImageNet dataset learn the next level of features from the data and link to the trainable parameters. These pre-trained weights won't be altered during the training of the hybrid model i.e. the weights are fast-frozen but can be fine-tuned.

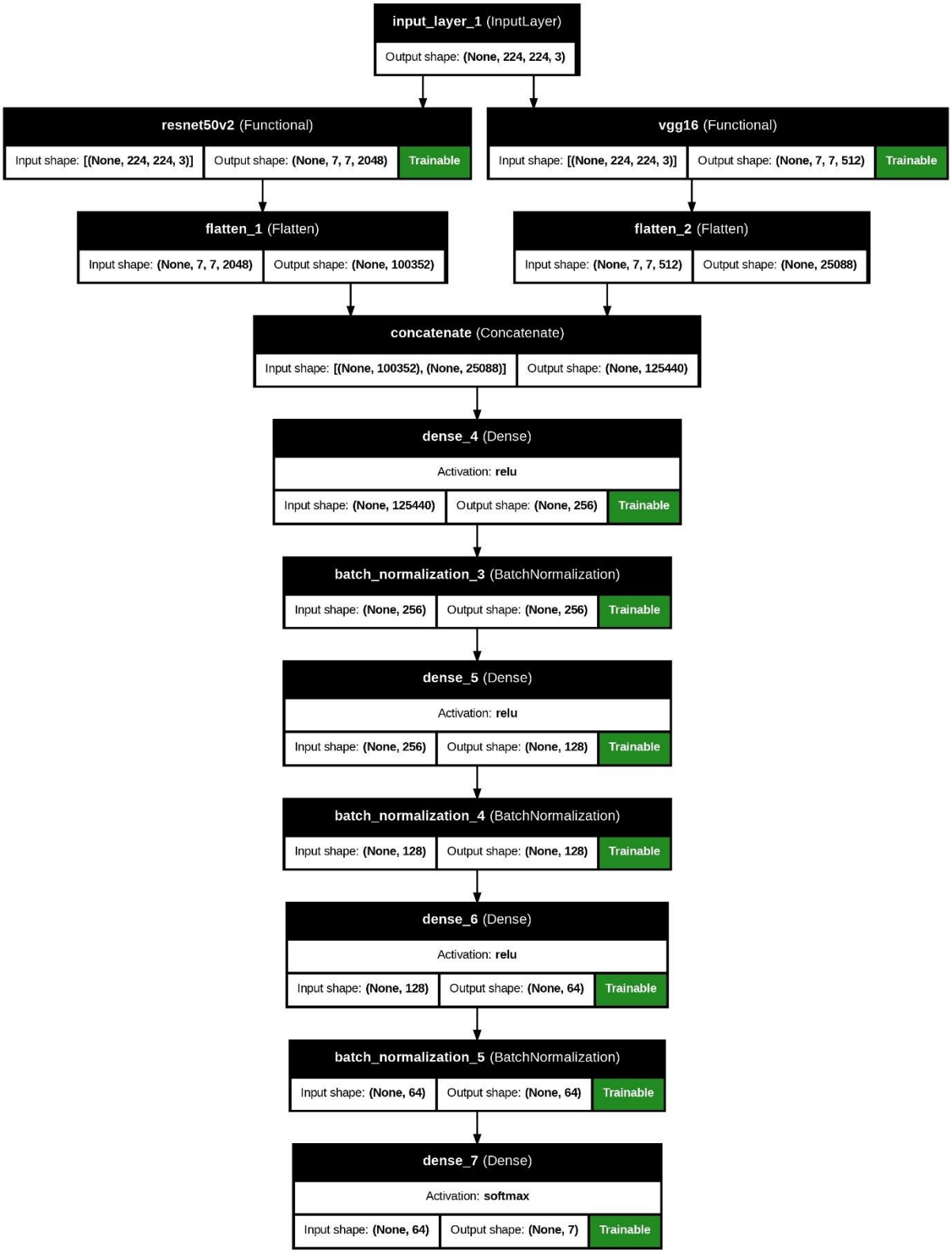


Figure 3-11Architecture of Vgg16 and resnet50v2 hybrid model transfer learning

3.4 Model Training Strategies and the Role of Callback Functions

One of the most challenging parts of training a deep learning model, especially for emotion detection from facial expression images, is figuring out the right architecture and also tuning the training process, where performance is maximized, while computational resources are minimized. Training strategies for the emotion detection model will be discussed in this section and we will be emphasizing the minimisation of computational resources and maximisation of validation accuracy. The model was trained sequentially on the Vietnamese-German University Emotion Recognition Dataset (VGU) and the IIMI Emotional Face Dataset (IIMI), then retrained on the FER dataset and importance of callback functions that were used during training for maximising performance and save the computing Resources. i.e. Early Stopping, Model Checkpoint, and ReduceLROnPlateau.

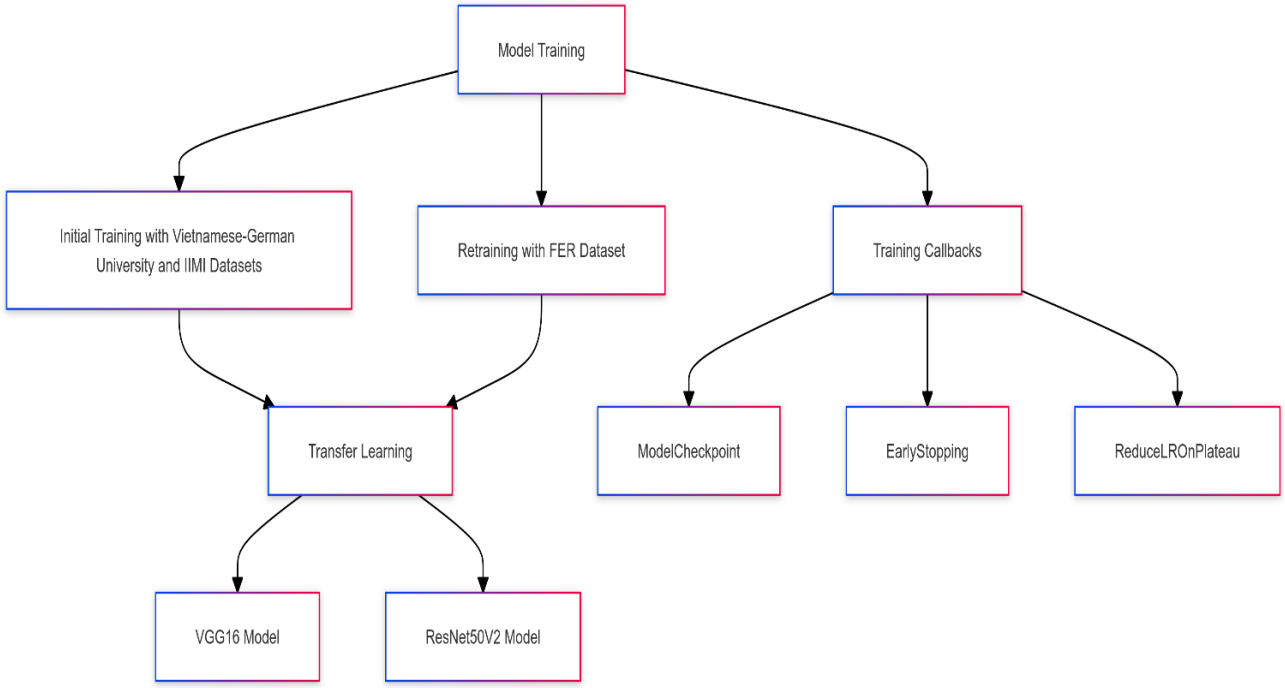


Figure 3-12 Flow Chart of Model training Strategy

3.4.1 Sequential Training on Multiple Datasets

Sequential training of the deep learning model on the many datasets also enables more generalisation. That is to say, first train the model with one dataset and then fine-tune or retrain it with the others. This was partly because of the diversity of the datasets in this study.

3.4.1.1 The Vietnamese-German University Emotion Recognition Dataset (VGU) and IIMI Emotional Face Dataset (IIMI):

The VGU dataset seemed to provide a good response to all the flaws the other datasets had. Firstly, it provided faces with a broad range of expressions. Training the model on VGU and IIMI would allow it to grasp more general facial features.

3.4.1.2 FER 2013 Dataset:

The last step was to further fine-tune the model on the FER dataset, which is the gold standard in terms of capturing different facial expressions. This was crucial for improving the model's generalisation across different populations and under different lighting conditions, and, consequently, to improve its performance on real-world data.

This sequential strategy for training allowed the model to self-improve by continually adapting to a variety of diverse learning environments, and to better generalise on variants of data it hasn't seen.

3.4.2 Computational Resource Management

Here, the key to successful deep learning was the judicious use of computational resources, since the model may have to work on large datasets with huge models. The e-model was trained in such a way that it kept the use of GPU to a minimum while maximising the validation accuracy (ie, accuracy after completion of training). The following measures were undertaken to achieve this.

3.4.3 Importance of Callback Functions

Callback functions are important as they control different aspects of the training of deep learning models. In this study, there are three types of callback functions that speed up and improve the training of the mode like Early Stopping, Model Checkpoint, and ReduceLROnPlateau. Each of the callback functions is carefully designed to enhance the training process and reduce the Computing resources while training and negate the issue of overfitting.

3.4.3.1 Early Stopping Callback

The Early Stopping callback monitors a given metric during training, such as the validation accuracy or validation loss, and stops this training process when the model stops improving on the validation set over a certain number of epochs. Not only does it prevent overfitting, it saves time and energy by halting training when the model is performing as well as it can on the validation set.

Impact on Training:

- Avoidance of overfitting: Early Stopping helped to avoid overfitting by stopping training as soon as the model was no longer improving on the validation set. Overfitting is a general problem with deep learning, where a model would become too attuned to the training data and perform poorly if presented with a new data set. Training can be stopped at the right moment so that the model keeps good generalisation properties.
- Resource Efficiency: Using Early Stopping was also more resource-efficient than training to the end of a predefined epoch, because it used fewer GPU hours of the large GPU we had been assigned for the experiment. Training a deep learning model can be very expensive, and stopping early can save significant compute resources. This was particularly helpful here because the experiment had a GPU quota.

3.4.3.2 Model Checkpoint Callback

The Model Checkpoint callback is vital for saving the best performing model at any point during training. With this callback, you specify an evaluation metric, and the

model weights get saved whenever there is an improvement. As a result, you know you'll have the model with the best performing set of weights, even if training continues past the point of optimal performance.

Impact on Training:

- Optimal Model Preservation: By saving the model weights only when the validation loss improved, the Model Checkpoint callback ensured that the best version of the model was saved – in cases in which models often overfit after some time, you can avoid retraining the model entirely, because the best model is already there.
- Avoid overfitting: Like the Early Stopping callback, the Model Checkpoint callback also helped to avoid overfitting, since the callback focused on validation loss. Even if the training loss kept decreasing, which could indicate overfitting, the validation loss still provided a better measure of how the model performs on new data. We saved the model at the point of lowest validation loss, ensuring that the final model was well-generalized.

3.4.3.3 ReduceLROnPlateau Callback

The ReduceLROnPlateau callback, which can be used to adjust the learning rate during training, should also be a part of your toolkit. The learning rate is the size of the steps the model takes in its weight updates. Set it too high, and the model might overshoot the global minimum; set it too low, and the model might take too long to converge or get stuck in a local minimum. The ReduceLROnPlateau callback will decrease the learning rate dynamically when the model's performance plateaus. It also allows more fine-tuned adjustments of the learning rate.

- Eased Convergence: When the validation loss stalled, the model started reducing the learning rate. This enabled the model to tweak the weights more accurately, easing the convergence. This is an important step towards better performance during the later stages of training when the large learning rates do not enable the model to reach its optimum.
- Efficient Resource Utilisation: The learning rate callback called ReduceLROnPlateau allowed us to train the model faster and more efficiently, without needing to waste resources on multiple training epochs at a suboptimal learning rate. At each learning rate change, new data points are represented more efficiently on the small scatter plot of learning rate vs validation loss.
- Preventing Training Stalls: The 'min_lr' prevented the learning rate from getting too small which could have caused the training to slow down or even stop. The compromise between these two effects ensured that the model could still get better without unnecessary delays, which facilitated efficient training.

3.5 Model Evaluation (Categorized the Emotions)

Evaluating the model is an important part of an emotion detection. It helps us to report on how the model performs at classifying emotions. Here we are using an emotion taxonomy that categorizes emotions into: Negative, Positive and Neutral.

The emotions have been aggregated in this manner, making the analysis easier to understand and containing only three categories. It helps us to understand whether we are correctly distinguishing between types of emotional valence.

To evaluate the model, we will use a confusion matrix and a classification report, both modified to reflect our three-category system.

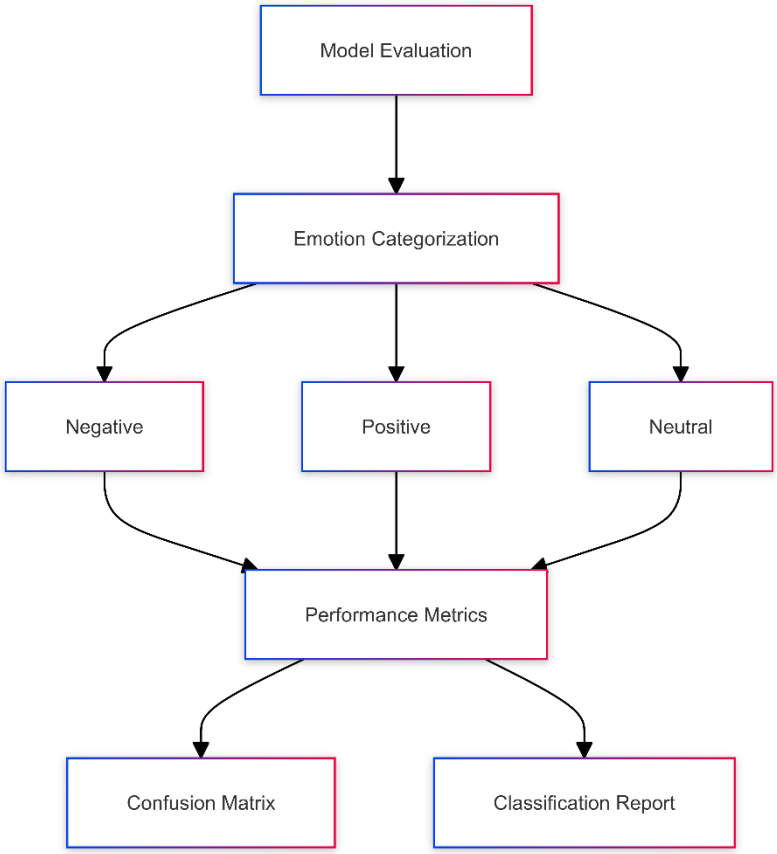


Figure 3-13 Flow Chart of Model Evaluation Process

3.5.1 Emotion Categorization

The emotions have been grouped into three categories based on their emotional valence:

- Negative Emotions: Angry, Disgusted, Fearful, Sad
- Positive Emotions: Happy, Surprised
- Neutral Emotion: Neutral

It also makes it easier to evaluate since we have important real-world uses in mind – for example, knowing the general valence (good/bad) of a situation would be useful, and might be of more relevance than being able to identify fear versus sadness.

3.5.2 Confusion Matrix

This is why it is very useful to compute a confusion matrix to get a summary of how the classification model performs. The confusion matrix compares the predicted labels of the classification model with the actual labels (ground truth). There are two types of confusion that we would like to avoid: false positives and false negatives. In the case of the emotion-

recognition task, false positives are outbursts of anger detected when the speaker is actually calm. False negatives would be the opposite scenario: the classification model fails to detect an outburst of anger when one exists. For the handshake-classification study, the confusion matrix contains the predicted labels categorised according to the three emotion categories.

	Predicated: Negative	Predicated: Positive	Predicated: Neutral
Actual: Negative	TP (Negative)	FP (Positive)	FN (Neutral)
Actual: Positive	FN (Negative)	TP (Positive)	FP(Neutral)
Actual: Neutral	FN(Negative)	FP (Positive)	TP (Neutral)

These are the four cells in the confusion matrix, giving the number of each possible actual/predicted emotion pair:

- True Positives (TP): The model correctly predicts the emotion category (it predicts 'negative' when the emotion actually is negative)
- False Positives (FP): The model mistakenly predicts an emotion category, for example, classification as 'positive' when the true emotion is negative.
- False Negatives (FN): The model doesn't predict the right emotion category (ie, guesses 'neutral' when the emotion is negative).

3.5.2.1 Interpretation of the Confusion Matrix:

- TP (Negative): Indicates how many times the model correctly identified a negative emotion.
- FP (Positive): Number of times where the model incorrectly classified a negative emotion as positive.
- FN (Neutral): Represents instances where a negative emotion was incorrectly classified as neutral.

3.5.3 Classification Report

The classification report in the lower portion of the screen shows the performance of the model for the three emotion categories in detail using the key metrics of: precision (recall) and F1-score; and support.

3.5.3.1 Precision:

- Definition: Precision is the fraction of true positive observations correctly predicted, divided by the total positive observations predicted (TP / (TP + FP)). Precision indicates the effectiveness of positive predictions.

What does 'precision is high' mean in Emotion Detection? Let's say a model makes 1,000 positive predictions, but there are only 100 positive examples in the gold standard. In that case, his precision is still high because most of his positive predictions were correct.

3.5.3.2 Recall:

- Definition: Recall is the proportion of positive observations that are correctly predicted to be positive (TP / (TP + FN)). It quantifies the model's ability to find every meaningful observation.

Interpretation of Recall in Emotion Detection: High recall for a category means the model is detecting most instances of that category. For example, high recall for 'negative' indicates it captures most negative-sounding sentences.

3.5.3.3 F1-Score:

- Definition: The F1-score is the harmonic mean of precision and recall (2 * (Precision * Recall) / (Precision + Recall)). This is a single metric that balances both precision and recall.

Interpretation of the F1-score: The F1-score is the harmonic mean of precision and recall. One can think of the F1-score as a good metric to see how well your model works if precision and recall have a big difference. A high F1-score for a category indicates that the model has good precision and recall.

3.5.3.4 Support:

- Definition: Support: Number of occurrences per emotion category in the dataset.

Illustration for Interpretation in Emotion Detection: explanation of the precision, recall and F1-score metrics, which is how many example were available per category.

3.6 Deployment

This trained model for analysing emotions was deployed to Gradio, an online platform designed using a Python library. When users visit the page, they are welcomed with an interface that allows them to have a user-friendly and accessible experience by uploading videos and obtaining detailed emotion analysis as output.

The deployment includes many outputs, such as line graphs, bar charts, pie charts as well as downloading a CSV file that contains the detailed emotion analysis data.

3.6.1 Video Input and Preprocessing

After a user loads a video, the first stage in the analysis is to segment it into individual frames. Since the model processes the video in real time or at least near-real time and get the timestamps right, the model works through the video one frame at a time. This offers enough temporal granularity to capture changes in emotion, while still being computationally tractable.

Classifier is an older model than more sophisticated ones like Multi-task Cascaded Convolutional Networks (MTCNN). Newer models are more accurate at face identification, especially when a face is partially occluded or in different lighting conditions. However, in this case the Haar Cascade classifier was sufficient because predictive speed was paramount, and limited resources were available. Second, every application would run on laptops with different hardware configurations (memory, processor, etc), and their ability to handle and process the video feed needed to be preserved. The Haar Cascade classifier struck a good balance in this case, as it provided reasonable accuracy at a low computational requirement.

3.6.2 Emotion Detection and Smoothing Techniques

After detecting the face in each frame, the cropped facial region is sent to the emotion recognition model, so that it can predict which emotion the person in that frame is exhibiting. When used in real-world applications, emotions rarely change every second; rather, they usually persist for several seconds or even more. In order to address this, I developed a smoothing algorithm that considers a sequence of frames, and not a single frame in isolation.

In particular, the smoothing algorithm looks at every 20 frames of a video (equivalent to 20 seconds of video) and outputs the emotion that was most commonly classified within that 20-frame sliding window. This smoothing allows the predicted emotions to be more consistent, allowing the emotions to last for more than a brief moment as is common with real-world emotional expressions. This smoothing mechanism assigns the most common emotion in the 20-frame window to each frame in that window. This approach helps to smooth out any transient misclassifications that could have happened due to noise or sudden and short facial expressions. This smoothing increases the robustness of the emotion detection algorithm in addition to creating a more accurate depiction of the time structure of the displayed emotions.

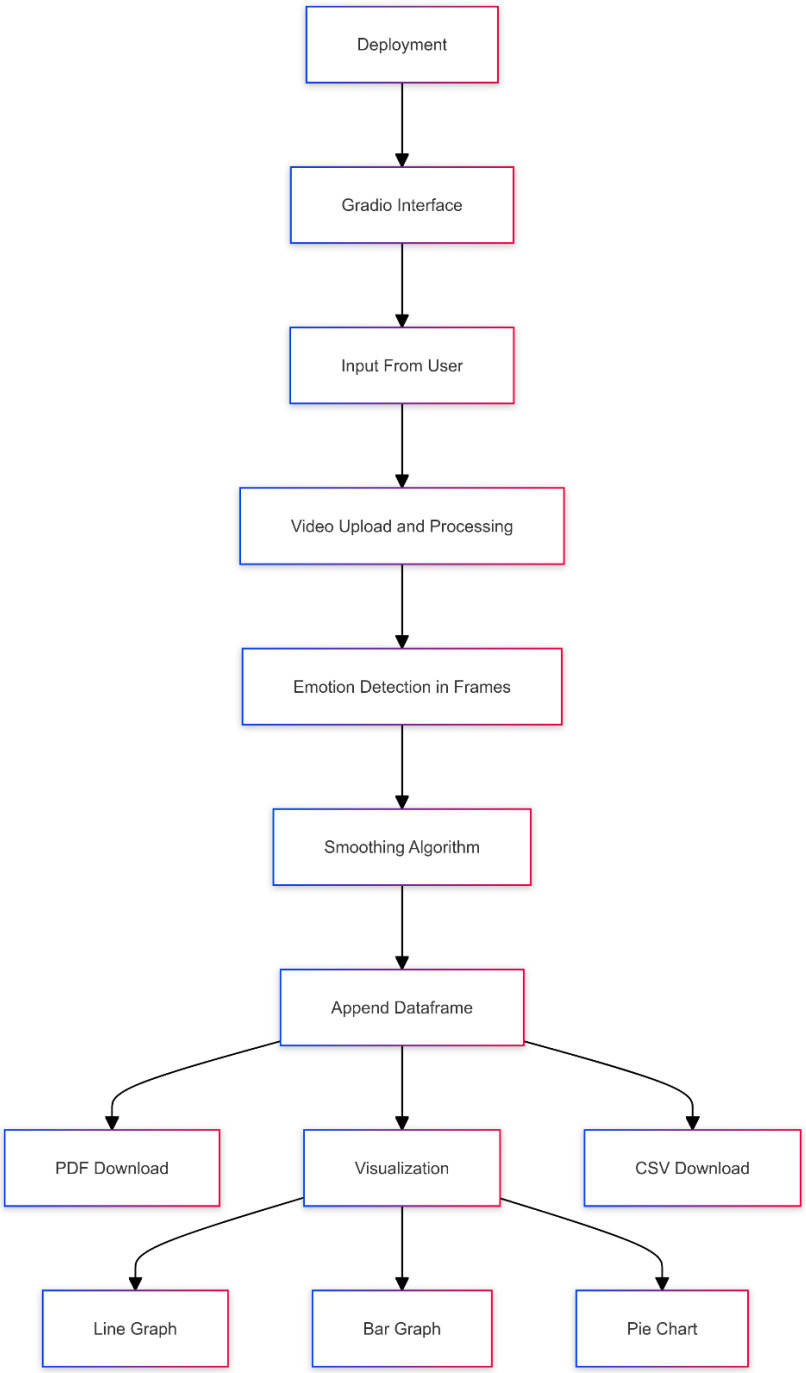


Figure 3-14 Flow Chart of Deployment Process

3.6.3 Visualization and Data Export

The deployment associates and generates several visualisations that help the user interpret the emotional information extracted from the video. The following visualisations are generated:

3.6.3.1 Line Graph of Continuous Emotion Detection

The graph denotes emotions detected over time. It is very temporal and can show the shift of emotion observed at a particular time range. Hence, the heightened changes in observed emotional states can be pinpointed, and information about the emotional dynamics in the video can be revealed. For example, if the video is a speech video, the graph might have areas of high anxiety, calmness or happiness while the keywords reveal what the speaker is talking about as he moves from one topic to the next.

3.6.3.2 Bar Graph of Emotion Distribution:

The bar graph displays the proportion of emotions roughly each minute of the video, allowing viewers to know the proportion of each emotion in the whole video. This is useful for understanding the overall emotional atmosphere of the video. If the video is filled with negative emotions, for instance, possibly it is a stressful or hard situation, but if there are plenty of positive emotions in the video, possibly it is a happy or lively event.

3.6.3.3 Pie Chart of Emotion Proportions:

As in the bar graph, in the pie chart the proportions of various kind of emotions are visualised which helps in comparing their relative frequencies precisely. The pie chart would be best for immediately understanding the emotional breakdown of the video in a visually friendly way.

3.6.3.4 CSV File for Further Analysis:

Alongside the visual outputs, the deployment also generates a CSV file containing the raw and smoothed emotion data for each frame – allowing users to aggregate the data across multiple videos or integrate it with other datasets. The CSV file turns the deployment into not just a tool for real-time visualisation, but also a data source that can be worked with in depth, after the event.

4 Implementation

4.1 Data Standardization

Sorting data into this structure allows its use in deep learning tasks. Apply the data standardisation to each dataset (details of running the code are given in the following section).

4.1.1 Vietnamese-German University Emotion Recognition Dataset

The Vietnamese-German University Emotion Recognition Dataset was delivered in a format not usually compatible with typical image classification model structures. As a result, a standardisation process was required, involving the following steps:

```python
import pandas as pd
import shutil
import os
# Load the CSV file
csv_file = r'data\train\aa.csv'  # replace with your actual CSV file path
data = pd.read_csv(csv_file)
# Print column names to verify they are correct
print("Column names in CSV:", data.columns.tolist())
# Strip whitespace from column names
data.columns = data.columns.str.strip()
# Define the source directory containing the images
source_dir = r'data\train'  # replace with your actual image directory
# Define the destination base directory
destination_base_dir = 'dataset/train'  # replace with your actual destination directory
# Ensure destination folders exist for each emotion
emotions = ['Anger', 'Contempt', 'Disgust', 'Fear', 'Happy', 'Neutral', 'Sad', 'Surprise']
for emotion in emotions:
    os.makedirs(os.path.join(destination_base_dir, emotion), exist_ok=True)
# Function to get the emotion from the row
def get_emotion(row):
    for emotion in emotions:
        if row[emotion] == 1:
            return emotion
    return None
# Process each row in the CSV
for index, row in data.iterrows():
    filename = row['filename']
    emotion = get_emotion(row)
    if emotion:
        source_path = os.path.join(source_dir, filename)
        destination_path = os.path.join(destination_base_dir, emotion, filename)
        if os.path.exists(source_path):
            shutil.move(source_path, destination_path)
        else:
            print(f"File {source_path} not found!")
print("Images have been sorted into folders based on emotions.")
```

Figure 4-1 Screenshot of Python Code for Data Standardization for VGU dataset

- Loading and Cleaning the CSV File: The provided dataset contained a CSV file containing the filenames and their labels. The first step is to load that CSV file

into a pandas DataFrame and to clean up the column names by removing any extra whitespace. This was needed so that our operations could refer to the column names down the line.

- create directories for the pictures in a way that will make it easier for me to process them later on, when I want to train my model. I create seven directories, one for each class (emotion), under dataset/train.
- For each image that was labeled with a certain emotion in the CSV file, we made a new folder if necessary and moved the image into that directory. For example, if the get_emotion() function identified that an image was labeled with 'surprise', it would move the image to the 'surprise/' directory.

This process ensured that the images were correctly categorized, making them ready for the model training phase.

4.1.2 IIMI Emotional Face Dataset

The IIMI Emotional Face Dataset had to go through a similar standardisation process, although the first hurdle was to extract emotions from the filenames, as follows:

- Directory:As with the VGU dataset, directories were created for each emotion under a main directory, finaldataset.
- Extracting Emotion Labels and Sorting Images : The emotion label was encoded in the filenames of the IIMI dataset. To extract the emotion label from the filename, I implemented a get_emotion_from_filename() function and move the images to the corresponding emotion folder.
- This made it so that every image could be correctly categorised, allowing for easy access while the training was taking place.

```
In [7]:
import os
import shutil
# Directory containing images
image_dir = 'googledrive-archive'
# Directory to save organized images
output_dir = 'finaldataset'
# List of emotions based on your naming convention
emotions = ['ANG', 'DIS', 'FER', 'HPY', 'NUT', 'SAD', 'SUR']
# Create folders for each emotion if they don't exist
for emotion in emotions:
    emotion_dir = os.path.join(output_dir, emotion)
    if not os.path.exists(emotion_dir):
        os.makedirs(emotion_dir)
# Function to get the emotion from the filename
def get_emotion_from_filename(filename):
    parts = filename.split('_')
    if len(parts) > 3:
        return parts[3]
    return None
# Iterate through the images in the directory
for image_name in os.listdir(image_dir):
    emotion = get_emotion_from_filename(image_name)
    emotion=emotion[0:3]
    if emotion in emotions:
        src_path = os.path.join(image_dir, image_name)
        dest_path = os.path.join(output_dir, emotion, image_name)
        shutil.move(src_path, dest_path)
print("Images have been organized into corresponding folders.")
```

Figure 4-2 Screenshot of Python Code for Data Standardization for IIMI dataset

4.1.3 FER-2013 Dataset

Standardisation of the FER-2013 dataset involved translating pixel-level data saved in a CSV file to image files in order to ensure compatibility of the dataset with the other datasets used in the project. The code and steps for this conversion are shared in the appendix.

4.2 Data Preprocessing Implementation

Data pre-processing is a vital step of machine learning in general and especially in emotion detection from facial images. The preprocessing pipeline used for this project includes: data validation, class weighting, normalisation, scaling, resizing and batching. Below is the description of the preprocessing steps followed by the code snippets.

4.2.1 Data Validation

Validating the data is the initial and significant step in every machine learning project, and it is highly important when dealing with data about emotions detected from facial images. It helps us understand if only the valid data is contained and used as training dataset, which is the key to the final models' capacities for classification and generalisation.

Below are the main activities happened in the process:

- Filter out all files that are not images.
- Sanitise all corrupted or unreadable images.
- Make sure all images are in the expected formats.

```python
# Define the list of acceptable image extensions
image_exts = ['jpeg', 'jpg', 'png']

# Path to the directory containing image classes and possibly other nested subdirectories
data_dir = '/content/drive/MyDrive/dataset/dataset8emo/train'

# Walk through all directories and files in the dataset
for root, dirs, files in os.walk(data_dir):
    for file in files:
        # Construct the path to the current file
        file_path = os.path.join(root, file)

        try:
            # Check the file type of the current file
            file_type = imghdr.what(file_path)

            # If the file extension is not in the allowed list, remove it
            if file_type not in image_exts:
                print(f'Image not in ext list {file_path}')
                os.remove(file_path)
            else:
                # Proceed to process the image if needed, for example, reading it with OpenCV
                img = cv2.imread(file_path)

        except Exception as e:
            # Print out the issue and the path of the problematic file
            print(f'Issue with file {file_path}. Error: {e}')
            # Optionally, remove files that cause exceptions
            os.remove(file_path)
```

Figure 4-3 Screenshot of Python Code for Data Validation

4.2.2 Class Weight Calculation

This imbalance in classes is another challenge commonly faced in emotion detection tasks. To help with this problem, class weights can be used to make sure the model treats all classes equally when training, as shown in the following code.Weight is calculated for each class and then a dictionary is produced for mapping each class index to a class weight:

```python
# Extract class labels for all instances in the training dataset
classes = np.array(train_generator.classes)

# Calculate class weights to handle imbalances in the training data
# 'balanced' mode automatically adjusts weights inversely proportional to class frequencies
class_weights = compute_class_weight(
    class_weight='balanced',  # Strategy to balance classes
    classes=np.unique(classes),  # Unique class labels
    y=classes  # Class labels for each instance in the training dataset
)

# Create a dictionary mapping class indices to their calculated weights
class_weights_dict = dict(enumerate(class_weights))

# Output the class weights dictionary
print("Class Weights Dictionary:", class_weights_dict)
```

Figure 4-4 Screenshot of Python Code for Class Weight Calculation

4.2.3 Data Normalization and Scaling

4.2.3.1 Pixel Value Normalization

Normalisation is important to make sure that the model works with the input in the same way.Pixel values were normalized from the range [0,255] to [0,1]. This helps to normalize the input data and make the training process more stable.

```python
import tensorflow as tf

# Define normalization function
def normalize_img(image, label):
    image = tf.cast(image, tf.float32)  # Convert image to float32
    image = image / 255.0  # Normalize pixel values to [0, 1]
    return image, label
```

Figure 4-5 Screenshot of Python Code for Pixel Value Normalization

4.2.3.2 Data Scaling and Resizing

The images were resized to 224 x 224 pixels to make the input data uniform. Doing so impacts learning by reducing the computational complexity. The process is demonstrated with the code snippet that shows how the datasets were loaded for the given image size and batch size.

```python
# Load datasets
batch_size = 64
image_size = (224, 224)

train_dataset = tf.keras.utils.image_dataset_from_directory(
    directory=train_dir,
    labels="inferred",
    label_mode="categorical",
    color_mode="rgb",
    batch_size=batch_size,
    image_size=image_size,
    shuffle=True,
    seed=123,
    validation_split=0.2,
    subset="training",
    interpolation="bilinear",
    crop_to_aspect_ratio=False
)

validation_dataset = tf.keras.utils.image_dataset_from_directory(
    directory=train_dir,
    labels="inferred",
    label_mode="categorical",
    color_mode="rgb",
    batch_size=batch_size,
    image_size=image_size,
    shuffle=True,
    seed=123,
    validation_split=0.2,
    subset="validation",
    interpolation="bilinear",
    crop_to_aspect_ratio=False
)

test_dataset = tf.keras.utils.image_dataset_from_directory(
    directory=test_dir,
    labels="inferred",
    label_mode="categorical",
    color_mode="rgb",
    batch_size=batch_size,
    image_size=image_size,
    shuffle=False,
    interpolation="bilinear",
    crop_to_aspect_ratio=False
)
```

Figure 4-6 Screenshot of Python Code for Data Scaling,Resizing and Batching Process

Here, use of ' tf.keras.utils.image_dataset_from_directory' because it is consistent with the way handle the data, and because it is consistent with the previous version (ImageDataGenerator) used that handles the image batching process for this project

This preprocessing pipelining guarantees that the data is ready to undergo training for training of the models, maintaining the proportionality of classes, normalising and scaling the images, and handling the data via a combination of efficient resizing and batching. For a more detailed account of these preprocessing steps, please refer to the methodology section.

4.3 Model Selection and Experimentation Implementation

Three different emotions detection architectures were implemented: VGG16, ResNet50V2 and a hybrid of the VGG16 and ResNet50V2 models. The models we chose addressed the requirements to address emotions classification. Each model mentioned will be explained in detail, reason for using the pre-trained weights, the reason for using the layer freezing technique, reason for using the functional model type over the sequential model type.

4.3.1 Hybrid Model Architecture Implementation Explanation

In the hybrid model we use two powerful deep learning libraries provided out of the box by TensorFlow and Keras: tensorflow.keras.applications, tensorflow.keras.layers and tensorflow.keras.models. These libraries provide pre-built models and tools that ease the process of assembling complex neural networks.

```python
from tensorflow.keras.applications import ResNet50V2, VGG16
from tensorflow.keras.layers import Input, Flatten, Dense, BatchNormalization, Concatenate
from tensorflow.keras.models import Model

def Create_Combined_Model():
    # Define the input tensor
    inputs = Input(shape=(224, 224, 3))

    # Load the pre-trained ResNet50V2 model without the top layer
    resnet50v2_base = ResNet50V2(input_tensor=inputs, include_top=False, weights='imagenet')

    # Freeze all layers except the last 50 layers in ResNet50V2
    resnet50v2_base.trainable = True
    for layer in resnet50v2_base.layers[:-50]:
        layer.trainable = False

    # Load the pre-trained VGG16 model without the top layer
    vgg16_base = VGG16(input_tensor=inputs, include_top=False, weights='imagenet')

    # Freeze all layers except the last 5 layers in VGG16
    vgg16_base.trainable = True
    for layer in vgg16_base.layers[:-5]:
        layer.trainable = False

    # Get the outputs of both models
    resnet50v2_output = resnet50v2_base(inputs, training=True)
    resnet50v2_output = Flatten()(resnet50v2_output)

    vgg16_output = vgg16_base(inputs, training=True)
    vgg16_output = Flatten()(vgg16_output)

    # Concatenate the outputs of both models
    concatenated = Concatenate()([resnet50v2_output, vgg16_output])

    # Add custom layers on top of the concatenated outputs
    x = Dense(256, activation='relu')(concatenated)
    x = BatchNormalization()(x)
    x = Dense(128, activation='relu')(x)
    x = BatchNormalization()(x)
    x = Dense(64, activation='relu')(x)
    x = BatchNormalization()(x)
    outputs = Dense(7, activation='softmax')(x)

    # Create the model
    model = Model(inputs=inputs, outputs=outputs)

    return model
```

Figure 4-7 Screenshot of Python Code for Hybrid Model Architecture

4.3.1.1 Input Layer

Input Tensor (from tensorflow.keras.layers): Input (shape=(224, 224, 3)) This tells the network that the input of the model will be in the form of an image – an array of 224x224 pixels, with 3 colour channels of RGB. Notice that the Input layer has no weights learnt, and also no non-linearity activation. The entire network is built on top of this initial layer.

4.3.1.2 VGG16 Branch

VGG16 Base Model (VGG16 from tensorflow.keras.applications): The VGG16 model is loaded using **VGG16(input_tensor=inputs, include_top=False, weights='imagenet').** The include_top=False argument ensures that the fully connected layers at the top of the VGG16 model are excluded, allowing us to customize the top layers for our specific task. The weights='imagenet' argument loads pre-trained weights from the ImageNet dataset, which are useful for general feature extraction.

- Layer Freezing: All the layers of VGG16 up until the fifth last layer are frozen to preserve pre-trained features, i.e. the layers in vgg16_base represent the entire net (except the final fully connected layer) and using the loop we iterate through vgg16_base.layers and **set vgg16_base.layers[i].trainable = False** for all **i except the last five.** This way, the last five layers are what will be tuned during training while all the other layers remain constant.
- Flatten Layer (from tensorflow.keras.layers): Flatten converts the multi-dimensional feature maps at the outputs of convolutional layers to a 1D vector for fully connected layers to take input.

4.3.1.3 ResNet50V2 Branch

ResNet50V2 Base Model (ResNet50V2 from tensorflow.keras.applications):Likewise, we load ResNet50V2 as ResNet50V2(input_tensor=inputs, include_top=False, **weights='imagenet')**. ResNet50V2 the residual connections help train deep networks by overcoming the vanishing gradient problem.

- Layer Freezing: A loop through **resnet50v2_base.layers sets layer.trainable = False** for every layer up to the last 50. This allows fine-tuning of the last 50 layers, while retaining features learned from ImageNet.
- Flatten Layer (Flatten from tensorflow.keras.layers): The output from ResNet50V2 is also flattened using Flatten() to prepare the features for the fully connected layers.

4.3.1.4 Concatenation Layer

Concatenation (Concatenate from tensorflow.keras.layers): The outputs from the VGG16 and ResNet50V2 branch are flattened and concatenated using Concatenate()([resnet50v2_output,vgg16_output]). It is a weighted combination of the resnet50v2_output and vgg16_output and the concatenate layer as a combination of outputs from the two different network classes. In other words, the model will have access to a more extensive feature space for classification.

4.3.1.5 Fully Connected Layers

Dense Layers (from tensorflow.keras.layers.Dense): After concatenation, the produced feature vector goes through multiple Dense layers. These layers are often referred to as 'fully connected layers' and represent the neuron layer perceiving the input features and transform them into a more compact representation for classification.

- first Dense Layer: Dense(256, activation='relu') reduces the dimensionality and adds non-linearity with the ReLU activation function.
- Second Dense Layer: Dense(128, activation='relu') further refines the feature set.
- Third Dense Layer: Dense(64, activation='relu') continues to reduce the dimensionality until it's ready for the final classification layer.
- Batch Normalisation (tensorflow.keras.layers.BatchNormalization) – After each Dense layer, there is a BatchNormalization() to normalise inputs of the subsequent layer, to help stabilise and speed up training.
- Output Layer (Dense with softmax activation): This is the last layer and it is Dense(7, activation='softmax'), where 7 refers to the number of emotions classes, and the softmax activation function transforms the logits to probabilities for each class, and this helps in performing multiclass classification.

4.3.1.6 Model Compilation

The model is compiled with Adam optimiser **('adam'),** categorical cross entropy **('categorical_crossentropy')** for loss, and accuracy **('accuracy')** for performance measuring. This is how it is normally done for classification tasks with more than two outputs as it ensures that the model is optimised to correctly classify any of the many possible outputs.

4.3.2 VGG16 and ResNet50V2 Architectures Implementation

The main task was to build architectures for both VGG16 and ResNet50V2 models, in which they were developed by loading the base models without their top layers and after that added the custom fully connected layers on their top. Fully connected layers such as Flatten, Dense and Batch Normalization were used for both pre-trained networks in the way that they were used before in the hybrid model to fine tune the pre-trained networks in emotion detection task. For more details of these implementations are given in the appendix.

4.3.3 Use of the Functional Model API

The functional model API can handle both linear topology and non-linear topology (with shared layers and multiple inputs or outputs), it was used to define models regardless of the underlying architecture. Although the VGG16 model is a sequential model with a flat linear topology, the functional API was used here for consistency of how models are defined and so that structural changes can be easily made in the future without having to convert the model's definition to the functional API.

ResNet50V2 and the hybrid required the functional API for the same reason: their models included residual connections and concatenation of the outputs of multiple branches. While these both added more complexity to the pretrained models, the functional API also means that have greater control of model architecture; this is important if combine pretrained models, like ResNet50V2, with other models, or use more advanced techniques, such as layer freezing selectively.

For detailed information about the parameters used and the experimental steps, please refer to the methodology section.

4.4 Model Training Implementation

In the implementation phase of the model training process, specific techniques were employed to optimize training performance while maintaining high validation accuracy and minimizing computational resource usage. The detailed steps and rationale behind the training process are explained in the methodology section. This section focuses on the use of callbacks and the application of AUTOTUNE to enhance the efficiency of the training process.

4.4.1 Callbacks Implementation

Callbacks are an essential part of the training process, offering mechanisms to control and optimize the training dynamics based on real-time feedback from the model's performance on the validation set. The following callbacks were implemented:

```
# File path for the model checkpoint
cnn_path = '/content/drive/MyDrive/Emotion_Detection27/VGG16'
name = 'mymodelferwithold.keras'
chk_path = os.path.join(cnn_path, name)

# Callback to save the model checkpoint
checkpoint = ModelCheckpoint(filepath=chk_path,
                             save_best_only=True,
                             verbose=1,
                             mode='min',
                             monitor='val_loss')

# Callback for early stopping
earlystop = EarlyStopping(monitor = 'val_accuracy',
                          patience = 8,
                          restore_best_weights = True,
                          mode = 'max',
                          verbose = 1
                          )

# Callback to reduce learning rate
reduce_lr = ReduceLROnPlateau(monitor='val_loss',
                              factor=0.2,
                              patience=2,
                              min_lr=0.0005,
                              mode='min',
                              verbose=1
                              )

# Callback to log training data to a CSV file
csv_logger = CSVLogger(os.path.join(cnn_path,'trainingF2.log'))

# Aggregating all callbacks into a list
callbacks = [checkpoint, earlystop,csv_logger]  # Adjusted as per your use-case
```

Figure 4-8 Screenshot of Python Code for Callbacks

Model Checkpointing (ModelCheckpoint):

- Objective: To save the model at the point where it performs best on the validation set.
- Implementation: The ModelCheckpoint callback was configured to save the model's weights to a specified file path (chk_path). The save_best_only=True parameter ensures that only the best model (based on minimum validation loss) is saved, preventing unnecessary storage of intermediate models. The monitor='val_loss' and mode='min' parameters ensure that the model is saved when the validation loss decreases.

Early Stopping (EarlyStopping):

- Objective: To halt training once the model stops improving, thereby preventing overfitting and reducing computational load.
- Implementation: The EarlyStopping callback was set to monitor val_accuracy, with a patience of 8 epochs. This means training stops if there is no improvement in validation accuracy for 8 consecutive epochs. Additionally, restore_best_weights=True was used to ensure that the model reverts to the best-performing weights at the end of training, rather than the weights from the last epoch.

Learning Rate Reduction (ReduceLROnPlateau):

- Objective: To adaptively reduce the learning rate when the model's performance plateaus, which can help in fine-tuning and avoiding local minima.
- Implementation: The ReduceLROnPlateau callback monitors val_loss and reduces the learning rate by a factor of 0.2 if there is no improvement for 2 epochs (patience=2). The min_lr parameter is set to 0.0005, ensuring that the learning rate does not fall below this threshold.

Training Logging (CSVLogger):

- Objective: To keep a record of the training process, including the loss and accuracy for each epoch.
- Implementation: The CSVLogger callback logs training and validation metrics to a CSV file located at the specified path. This log can be invaluable for post-training analysis and troubleshooting.

Aggregated Callbacks:

All the above callbacks were aggregated into a list, callbacks = [checkpoint, earlystop, csv_logger], ensuring they are all applied during model training.

4.4.2 AUTOTUNE for Optimized Data Loading

In addition to callbacks, the AUTOTUNE feature of TensorFlow was employed to optimize the data loading and preprocessing pipeline. This technique significantly enhances the efficiency of the training process by ensuring that data is fed into the model as quickly as possible, reducing idle time for the GPU.

Objective: To automatically tune the data loading and preprocessing operations for optimal performance, particularly in environments with constrained computational resources.

Implementation: The AUTOTUNE option is utilized in the data pipeline with the cache() and prefetch() methods, applied to the training, validation, and test datasets.

```
AUTOTUNE = tf.data.AUTOTUNE

train_dataset = train_dataset.cache().prefetch(buffer_size=AUTOTUNE)
validation_dataset = validation_dataset.cache().prefetch(buffer_size=AUTOTUNE)
test_dataset = test_dataset.cache().prefetch(buffer_size=AUTOTUNE)
```

Figure 4-9 Screenshot of Python Code for Autotune

Explanation:

cache(): Caches the dataset in memory, enabling faster access during training, especially useful when the dataset fits into memory.

prefetch(buffer_size=AUTOTUNE): Prefetches batches of data in the background while the current batch is being processed by the model. The AUTOTUNE buffer size allows TensorFlow to determine the optimal number of batches to prefetch automatically, balancing the load between data loading and model training.

By integrating these techniques, the model training process is not only made more efficient but also more adaptive to the complexities of real-time performance, leading to better overall results with reduced computational overhead. This approach ensures that the models are trained effectively on the given datasets, maximizing accuracy while minimizing the required resources. For further details on the training steps and parameters, please refer to the methodology section.

4.5 Practical Considerations in Deployment

There were several practical reasons behind these choices. First, the Haar cascade classifier (as opposed to MTCNN or any other more advanced model) was used so as not to use a lot of processing resources thus making it possible to run the deployment both on machines with limited processing power and forcing an end user to use it on the browser rather than installing additional software (and thus broadening the potential user base to people who do not have a high-end hardware, rich features and a powerful GPU).

Second, the use of a smoothing algorithm bespeaks practical, real-world utility. Real-world users of such programmes are more interested in empirically consistent, reliable emotion detection than unfiltered film of every micro-expression. The smoothing technique helps to filter out noise and relatively minor shocks to emotional states, reducing the variability of our inferences into quantifiable, repeatable forms that closely approximate those in real-world experiences.

User and their willingness to provide visual access to data for various purposes. If user needed only a quick grasp of the overall summary of the data, there were different views provided accordingly, the options to export the data made the deployment even more user-friendly and specific to needs. Possibilities of data exportation could facilitate the further

investigation and creation of reports with the freely accessible data. Additionally, the easy accessibility and applicability of such visualisation made this tool suitable for different purposes.

Deployment of the emotion analysis model using Gradio is highly responsive towards the demand of computational efficiency and requirement of accuracy and reliability in emotion detection. The application of the Haar Cascade model for detection of faces makes this deployment response more practical and accessible for video-based emotion analysis. Along with the availability of multiple visualisation options and opportunity to export data, this tool facilitates the user in understanding emotional dynamics in video to a great extent.

4.5.1 Deployment Stages

The key to making a video-based emotion analysis application useful and easy to use is the clever use of the Gradio interface, which allows users to upload videos and see the emotional analysis results. The application has been built modularly, carefully, and with consideration given to all the functional aspects, so that its capabilities and behaviors are intuitive, robust and informative enough to provide users with a comprehensive understanding about what the video file is about emotionally.

```
logo='university-of-huddersfield-239-logo.png'
# Create the Gradio interface
inputs = gr.Video(label="Upload Your Video",include_audio=True,min_width=550)
outputs = [
    gr.Image(label="Emotion Trends Over Time"),
    gr.Image(label="Bar Plot of Emotion"),
    gr.Image(label='Pie Chart of Emotion Categories'),
    gr.File(label="Download PDF Report",),
    gr.File(label="Download CSV Data")
]

# Create the Gradio interface
gr_interface = gr.Interface(
    fn=lambda video: process_and_display_video(video, model),
    inputs=inputs,
    outputs=outputs,
    article='This tool is developed by Dirdh Patel(U2366489) for MSc dissertation purposes at the University of Huddersfield Under the supervision of Dr.Tianhua Chen.',
    allow_flagging="never",
    live=True,  # Ensure the video is completely uploaded before processing
)

# Create a Gradio layout with custom CSS
with gr.Blocks() as demo:
    gr.Markdown(
        "<h1 style='text-align: center; margin-bottom: 32px;'>Emotion Recognition from Video</h1>"
    ),
    gr.Image(logo, show_label=False, interactive=False, height=165, show_download_button=False, show_fullscreen_button=False),

    gr.Markdown(
        "<h2 style='text-align: center; margin-bottom: 20px;'>Upload a video from Webcam to analyze the emotions displayed.</h2>"
    ),
    gr.Markdown(
        "<h3 style='text-align: center; margin-bottom: 20px;'>This tool will provide a detailed analysis including visualizations and a downloadable report.</h3>"
    ),
    gr.Markdown(
        "<h3 style='text-align: center; margin-bottom: 20px;color: blue;'>This tool is developed by Dirdh Patel(U2366489) for MSc dissertation purposes at the University of Huddersfield under the supervision of Dr.Tianhua Chen.</h3>"
    ),
    gr_interface.render(),


demo.launch(show_error=True)
```

Figure 4-10 Screenshot of Python Code for Gradio Interface Deployment

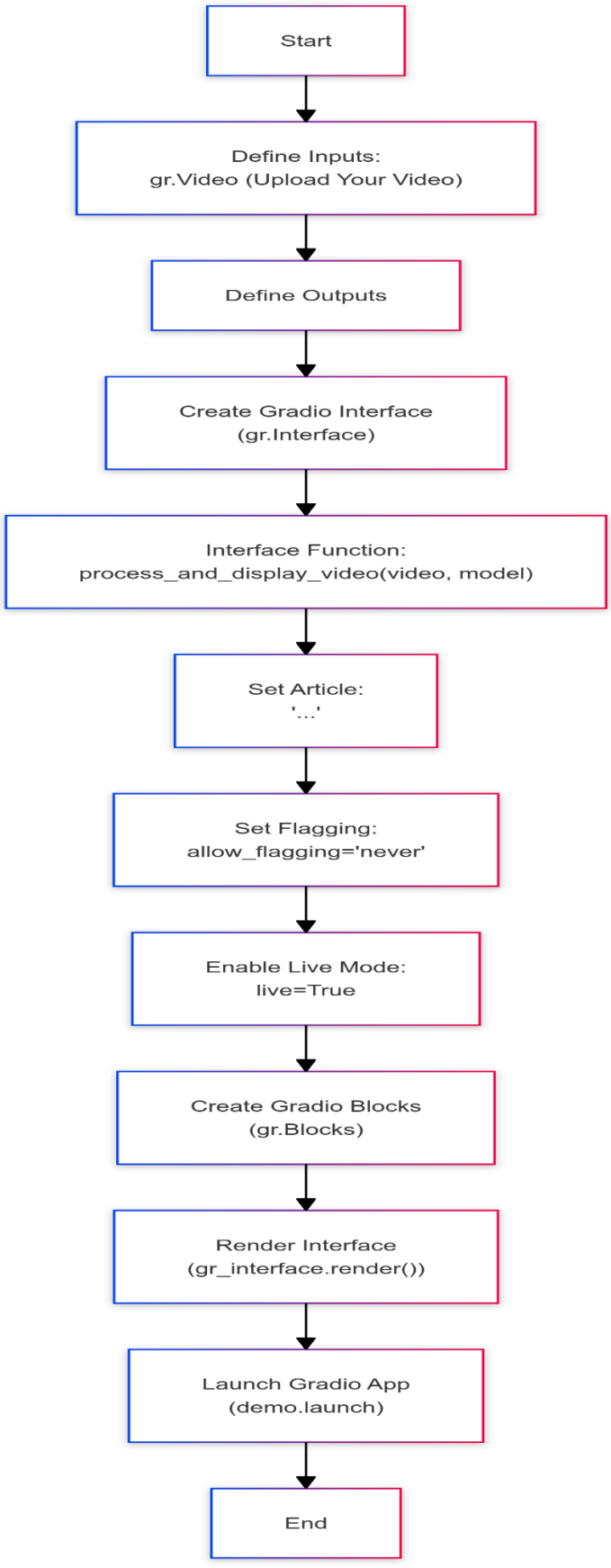


Figure 4-11 Flow Chart for Gradio Interface Deployment

4.5.1.1 Input Handling

The first stage of the app takes the user inputs, which are video files, and provides a nice interface for them. The Gradio gr.Video component handles this nicely for us. We can decide that the user can upload a video (with audio) with a minimum width, so the user sees that and knows that the video will be interpeted as such. The audio is preserved and the video appears at an appropriate size for the processing.

4.5.1.2 Processing the Video

Upon successful video upload, the core processing function, process_and_display_video, is triggered

A. process_and_display_video Function

This function acts as the pipeline where the video streams in, is processed, and results in emotion predictions as well as visualisations and downloadable reports.

- The function Initially checks if a video has been loaded by the user. If so, the URL of the video is extracted and returned; otherwise, it prompts the user to upload a video.
- Processing the Video: It calls to the process_video function, this one process the video and return a DataFrame (df) that contains the given timestamp, the predicted emotion, and its class (negative, neutral or positive) .
- Emotion Category Mapping: we map the emotion categories to numbers, -1 for negative emotions, 0 for neutral, and +1 for positive; this numerical mapping is essential in plotting the emotion trend over time.
- Line Plot of Emotion Trends: The function generates a line plot that shows the tendency of detected emotion categories and visualizes the tendency of emotional state over time as the sequence progresses. This is a plot that shows the changing trend of emotions over time.
- Bar Plot: The total duration of all the frames associated with a certain type of emotion (negative, neutral, positive) is summarized and plotted on a bar chart. This helps determine the length of time one emotion was persistent for the duration of the video.
- Pie Chart of Emotion Distribution: A pie graph is created to visualise the percentage distribution of different types of emotion categorized throughout the video. The graph indicates the proportion of time spent on each emotional state.
- Create PDF Report: A function creates a PDF report with all the visualisations rendered and can be downloaded for further exploration.
- Return: In the end, the function returns paths to its output images (line plot, bar plot, pie chart) as well as to its PDF output report and CSV file with detailed emotion analysis.

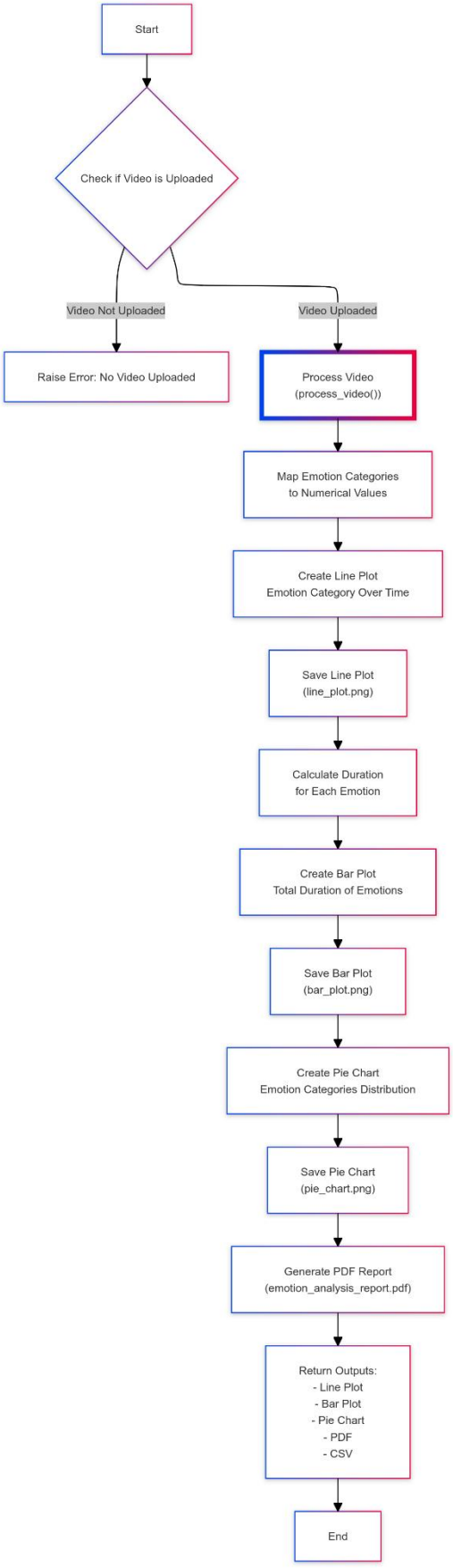


Figure 4-12 Flow Chart of Logic Behind Process and Display Function

```python
def process_and_display_video(video_path, model, smoothing_window=5):
    """
    Processes the video, generates predictions, and displays the results in a Gradio interface.
    Returns multiple outputs including plots and a downloadable PDF report.
    """
    # Check if the video is uploaded
    if video_path is None:
        raise ValueError("No video uploaded. Please upload a video before submitting.")
    # Process the video and get predictions
    df = process_video(video_path, model, output_csv="temp_output.csv", smoothing_window=smoothing_window)
    category_mapping = {
        'negative': -1,
        'neutral': 0,
        'positive': 1
    }
    # Apply the mapping to the DataFrame
    df['Category_Num'] = df['Category'].map(category_mapping)
    # Create the plot
    plt.figure(figsize=(10, 6))
    sns.lineplot(x='Time (s)', y='Category_Num', data=df)
    plt.title("Emotion Category Over Time")
    plt.xlabel("Time (Second)")  # Change to seconds if not converting
    plt.ylabel("Category")
    # Set custom y-ticks
    plt.yticks([-1, 0, 1], ['Negative', 'Neutral', 'Positive'])
    line_plot_path = "line_plot.png"
    plt.savefig(line_plot_path)
    plt.close()
    # Create a bar plot of emotion counts
    df['Duration'] = df['Time (s)'].diff().shift(-1).fillna(0)
    category_durations = df.groupby('Category')['Duration'].sum()
    plt.figure(figsize=(10, 6))
    sns.barplot(x=category_durations.index, y=category_durations.values, palette='viridis')
    plt.xlabel('Emotion')
    plt.ylabel('Total Duration (seconds)')
    plt.title('Total Duration of Each Emotion Category')
    plt.xticks(rotation=45)
    bar_plot_path = "bar_plot.png"
    plt.savefig(bar_plot_path)
    plt.close()
    # Create a pie chart of emotion categories
    category_counts = df['Category'].value_counts()
    plt.figure(figsize=(8, 8))
    plt.pie(category_counts, labels=category_counts.index, autopct='%1.1f%%', startangle=140)
    plt.title('Emotion Categories Distribution')
    pie_chart_path = "pie_chart.png"
    plt.savefig(pie_chart_path)
    plt.close()
    # Generate PDF report
    pdf_path = "emotion_analysis_report.pdf"
    generate_pdf(line_plot_path, bar_plot_path, pie_chart_path, pdf_path)
    # Return the video, DataFrame, line plot, bar plot, pie chart, CSV file, and PDF report
    return line_plot_path, bar_plot_path, pie_chart_path, pdf_path, "temp_output.csv"
```

Figure 4-13 Screenshot of Python Code for Process and Display Function

B. process video Function

This is the function that processes the video frame by frame, detects the faces, predicts the emotions, and smoothed out the predictions.

- Face Detection using Haar Cascades: Loaded the Haar cascade classifier for detection of faces. Used to detect faces in each frame of the video.
- Frame Processing: There is a read-frame-by-frame loop. Once in every few frames (depending on the frame interval), a face is identified and the cropped face image is passed to the model to predict the emotion.

- Emotion Prediction: The model predicts an emotion label for the face that it has detected. These emotion labels are mapped onto broader categories (negative, neutral, positive).
- Smoothing Predictions: To minimize the possibility that there is noise in the predictions (ie, the emotions could change drastically from one frame to another), we use a smoothing function. In other words, the mode of the last couple of predictions is used to ensure that emotion is classified as more stable across time frames.
- Data Output: The results, with timestamp, predicted emotion and the category, stored to a Data Frame and saved to CSV file.

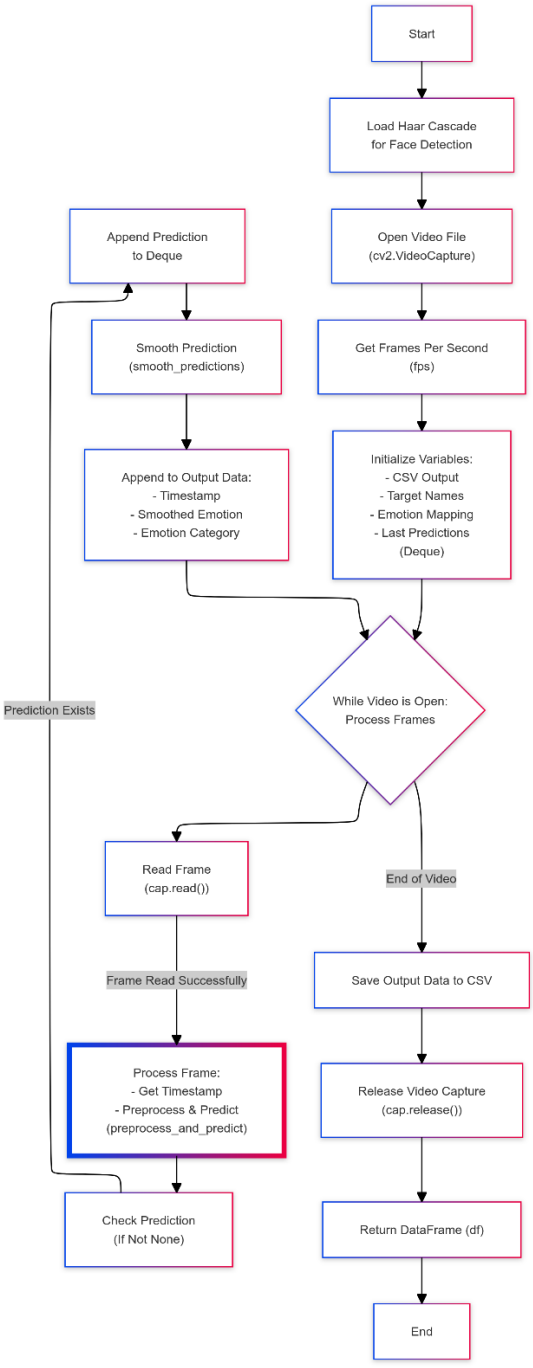


Figure 4-14 Flow Chart of Logic Behind Process Video Function

```python
def process_video(video_path, model, output_csv="output.csv", smoothing_window=5):
    """
    Processes a video, extracting frames at 5 frames per second,
    and saves the time and emotion predictions to a CSV file.

    Args:
        video_path: Path to the video file.
        model: The pre-trained Keras model.
        output_csv: Path to save the output CSV file.
        smoothing_window: Number of frames to consider for smoothing predictions.
    """
    # Load the Haar cascade for face detection
    face_cascade = cv2.CascadeClassifier(cv2.data.haarcascades + 'haarcascade_frontalface_default.xml')
    # Open the video file
    cap = cv2.VideoCapture(video_path)
    # Get frames per second (fps) of the video
    fps = cap.get(cv2.CAP_PROP_FPS)
    frame_interval = int(fps /2)
    # Prepare the CSV output
    output_data = []
    targetnames = ['angry', 'disgusted', 'fearful', 'happy', 'neutral', 'sad', 'surprised']
    # Define emotion mapping to categories
    emotion_mapping = {
        'angry': 'negative',
        'disgusted': 'negative',
        'fearful': 'negative',
        'sad': 'negative',
        'happy': 'positive',
        'surprised': 'positive',
        'neutral': 'neutral'
    }
    # Store the last few predictions for smoothing
    last_predictions = deque(maxlen=smoothing_window)
    frame_count = 0
    while cap.isOpened():
        ret, frame = cap.read()
        if not ret:
            break
        # Process the frame only if it's the correct interval
        if frame_count % frame_interval == 0:
            timestamp = frame_count / fps  # Get timestamp in seconds
            cropped_img, prediction = preprocess_and_predict(frame, model, face_cascade)
            if prediction is not None:
                predicted_class = np.argmax(prediction)
                emotion = targetnames[predicted_class]
                # Add prediction to the deque and smooth it
                last_predictions.append(emotion)
                smoothed_emotion = smooth_predictions(list(last_predictions))
                emotion_category = emotion_mapping[smoothed_emotion]
                output_data.append([timestamp, smoothed_emotion, emotion_category])
        frame_count += 1
    # Save to CSV
    df = pd.DataFrame(output_data, columns=["Time (s)", "Emotion", "Category"])
    df.to_csv(output_csv, index=False)
    # Release the video capture object
    cap.release()
    return df
```

Figure 4-15 Screenshot of Python Code for Process Video Function

C. Smooth_predictions Function

The smooth_predictions function helps stabilize the emotion predictions over consecutive frames.

- Window Size: The function uses the most recent predictions (up to size of the window) and returns the emotion with the highest frequency of predicted ratings. This short-term smoothing helps to attenuate spurious fluctuations of predictions, and provides a more realistic assessment of the emotional state over time.

```
def smooth_predictions(predictions, window_size=5):
    """
    Smooth predictions by using the mode of the last few predictions.

    Args:
        predictions: List of predictions to smooth.
        window_size: Number of frames to consider for smoothing.

    Returns:
        Smoothed prediction.
    """
    # Determine the actual window size to use
    actual_window_size = min(len(predictions), window_size)

    # Get the most common prediction in the last 'actual_window_size' frames
    most_common_prediction = Counter(predictions[-actual_window_size:]).most_common(1)[0][0]

    return most_common_prediction
```

Figure 4-16 Screenshot of Python Code for Smooth Prediction Function

D. preprocess_and_predict Function

This function preprocesses the frame of the video, then uses the emotion detection model to make a prediction.

- Grayscale Conversion: To turn the input video frame into grayscale. The reason is that face detection may have better performance on images of grayscale type.
- Face Detection: Using Haar cascade classifier, detect a rectangle that signifies the face. If a face is detected, bounding box of the face is extracted.
- Image Crop and Resize: A cropped face from the original frame is resized to 224x224 pixels, which is the input size for the emotion detection model.
- Cropping and resizing: the face image is cropped and resized to keep it at the same aspect ratio as the training images and the prediction is the same despite scale changes. 2. Normalisation: the input face image for a specific output class (also called label) is moved such that the average input value for all training images and the prediction is the same. This process also helps the model predict more accurately for any input value.
- Emotion Prediction: Take the preprocesses image input and obtain the predicted emotion of the person in the frame.

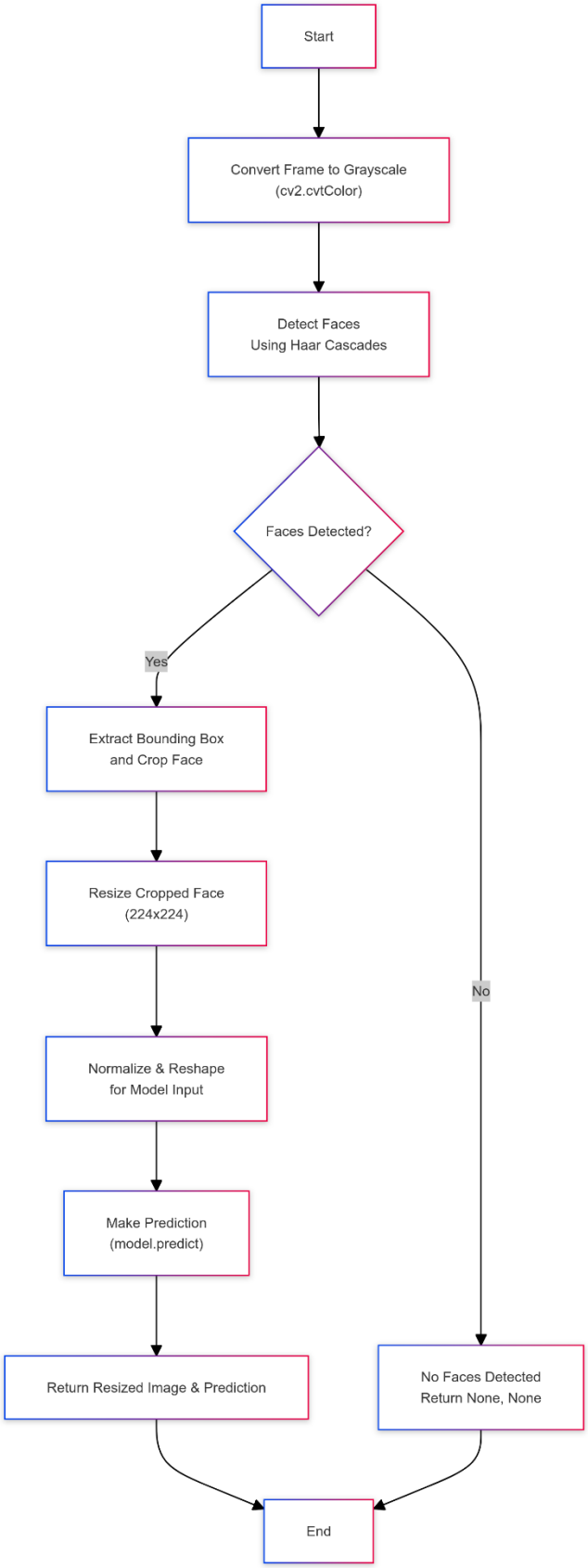


Figure 4-17 Flow Chart of Logic Behind Process and Predict Function

```python
def preprocess_and_predict(frame, model, face_cascade):
    """
    Detects a face using Haar cascades, crops it, resizes it,
    and predicts the class using the provided model.

    Args:
        frame: The video frame (numpy array).
        model: The pre-trained Keras model.
        face_cascade: The Haar cascade classifier for face detection.

    Returns:
        Tuple: (cropped_image, prediction)
    """

    # Convert frame to grayscale for face detection
    gray_frame = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)

    # Detect faces using Haar cascades
    faces = face_cascade.detectMultiScale(gray_frame, scaleFactor=1.1, minNeighbors=5, minSize=(30, 30))

    if len(faces) > 0:
        # Extract bounding box of the first detected face
        x1, y1, width, height = faces[0]

        # Crop the face
        cropped_img = Image.fromarray(frame[y1:y1 + height, x1:x1 + width])

        # Resize the cropped image
        resized_img = cropped_img.resize((224, 224))
        resized_array = img_to_array(resized_img)
        resized_array = resized_array / 255.0  # Normalize

        # Reshape for model input (assuming a single image prediction)
        input_array = np.expand_dims(resized_array, axis=0)

        # Make prediction
        prediction = model.predict(input_array)

        return resized_img, prediction
    else:
        return None, None
```

Figure 4-18Screenshot of Python Code for Preprocess and predict Function

4.5.1.3 Output Generation

After the video processing, a series of outputs are created to highlight the affective information :

- Emotion Trends Over Time : A line chart is generated to demonstrate how the recognized emotion changes over the video period of time. This helps users to follow their emotional journey and peaks while they are watching the video.

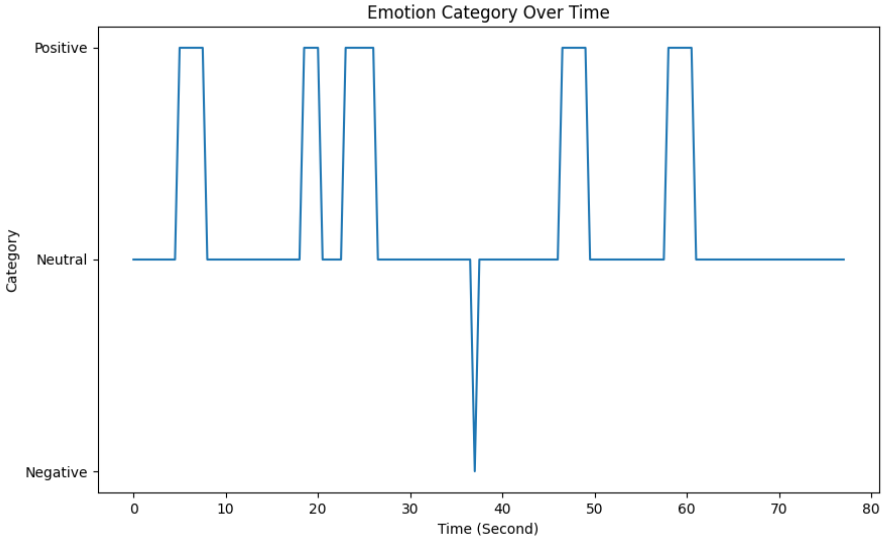


Figure 4-19 Sample Emotion over time Line Graph Generated By system

- Visualization of Emotions Using Bar Plots - We have a bar chart that illustrates the percentage of the detected emotions. In this visualization, it is concise to sum up the emotions that are closer to the y-axis.

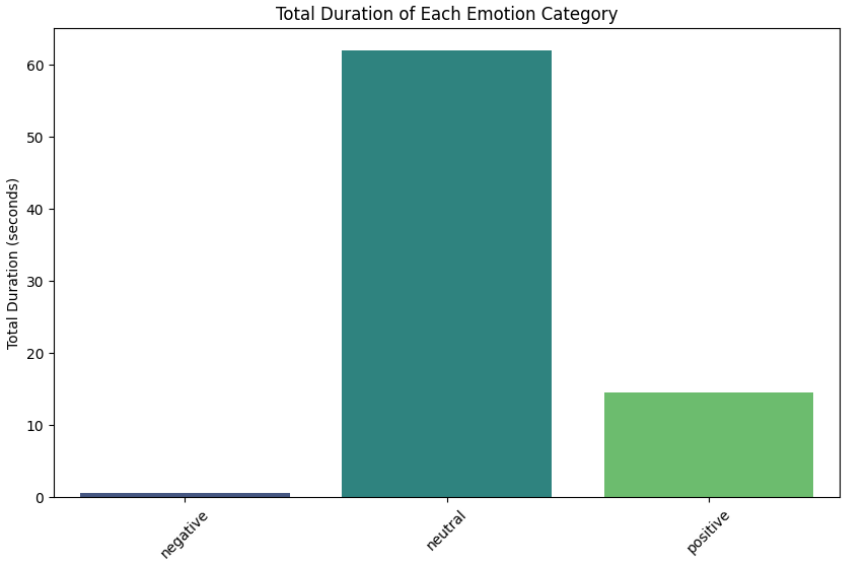


Figure 4-20 Sample Emotion Bar Chart Generated By system

- This pie chart demonstrates the proportion of emotion categories which are positive, negative and neutral.Overall, it is very easy to see a general overview of the distribution of emotions based on this pie chart. It is divided into three groups: positive, negative and neutral.

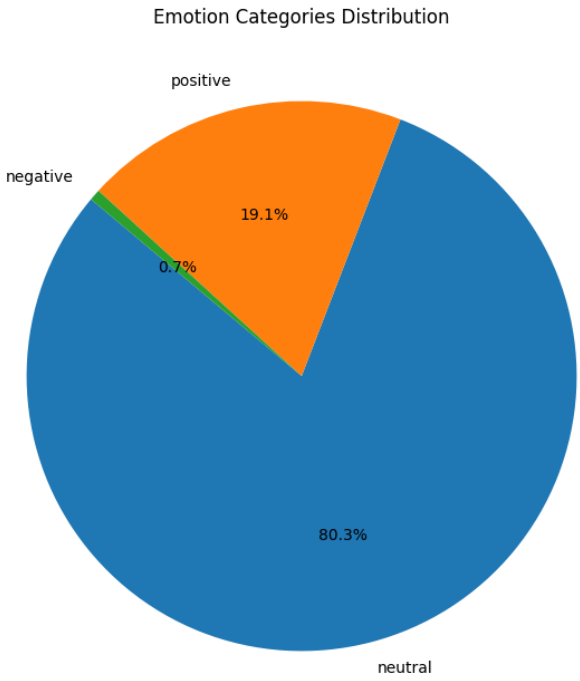


Figure 4-21 Sample Emotion Distrubution Pie Chart Generated By system

This pie chart can be used to indicate the distribution of emotions and illustrates that people feel negative emotions more than positive.

- The output also offers the ability to download a detailed PDF report plus a CSV file containing raw data. The downloadable reports are useful when further analysis is necessary or to keep track of those quotes.

This approach allows for a detailed, yet user-friendly, analysis of emotions in video content, making it a powerful tool for various applications such as psychological studies, media analysis, and human-computer interaction research.

5 Results, Analysis and Discussion

This chapter presents analysis for the performance of the models built for emotion recognition in this Project. The models are tested on the FER dataset. The classification reports and confusion matrices are discussed for the three different models, namely, the model built with VGG16 with transfer learning strategy, the model built with ResNet50 with fine tuning strategy and a hybrid model made by combining the best of practices from the previous two models. The last section concludes with discussion regarding the interface of the deployed model.

5.1 Model Performances

5.1.1 VGG16 Model with Transfer Learning

The VGG16 model, a well-known deep learning architecture, was utilized with transfer learning to classify emotions. The performance metrics are summarized in the classification report:

	precision	recall	f1-score	support
Negative	0.64	0.50	0.56	3340
Positive	0.46	0.77	0.58	2605
Neutral	0.31	0.05	0.09	1233
accuracy			0.52	7178
macro avg	0.47	0.44	0.41	7178
weighted avg	0.52	0.52	0.49	7178

Figure 5-1 Classification Report Of VGG16

The **VGG16 model** showed an overall **accuracy of 52%.** It did a fairly good job on the Negative class, having precision and recall of 0.64 and 0.50 respectively. Conversely, its showing was not that good on the Positive class, having precision of just 0.46 and a high recall of 0.77. Bad performance is demonstrated by the Neutral class emotions classification - poor precision score (0.31), recall score (0.05) and F1-score (0.09)

As shown in the confusion matrix below, there are instances where certain emotions were confused for each other, especially Neutral emotions, which were generally misclassified as Positive emotions.

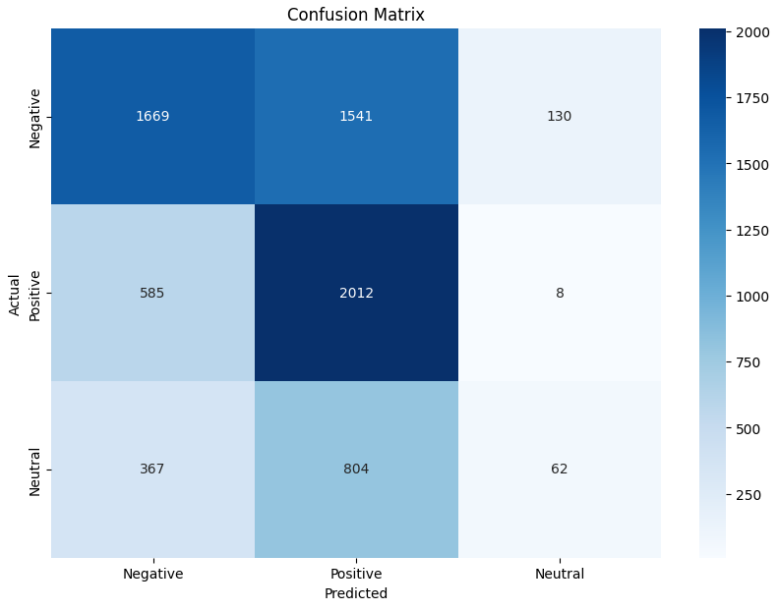


Figure 5-2 Confusion Matrix

5.1.2 ResNet50 Model with Transfer Learning

The ResNet50 model, which incorporates deeper layers and advanced residual learning, was fine-tuned for emotion recognition. The classification report is as follows:

	precision	recall	f1-score	support
Negative	0.81	0.78	0.80	3340
Positive	0.83	0.87	0.85	2605
Neutral	0.58	0.59	0.58	1233
accuracy			0.78	7178
macro avg	0.74	0.75	0.74	7178
weighted avg	0.78	0.78	0.78	7178

Figure 5-3 Classification Report Of Resnet50 model

The **ResNet50 model** significantly outperformed VGG16, achieving an overall **accuracy of 78%.** It excelled in recognizing both Positive and Negative emotions, with precision and **recall values above 0.80** for these categories. The Neutral class, while still a challenge, showed noticeable improvement compared to VGG16, with an F1-score of 0.58.

The confusion matrix for ResNet50 reveals fewer misclassifications, particularly in the Positive and Negative categories, showing the model's robustness in distinguishing between these emotional classes. This indicates that ResNet50's

deeper architecture and residual connections helped the model generalize better and extract more complex features from the input data.

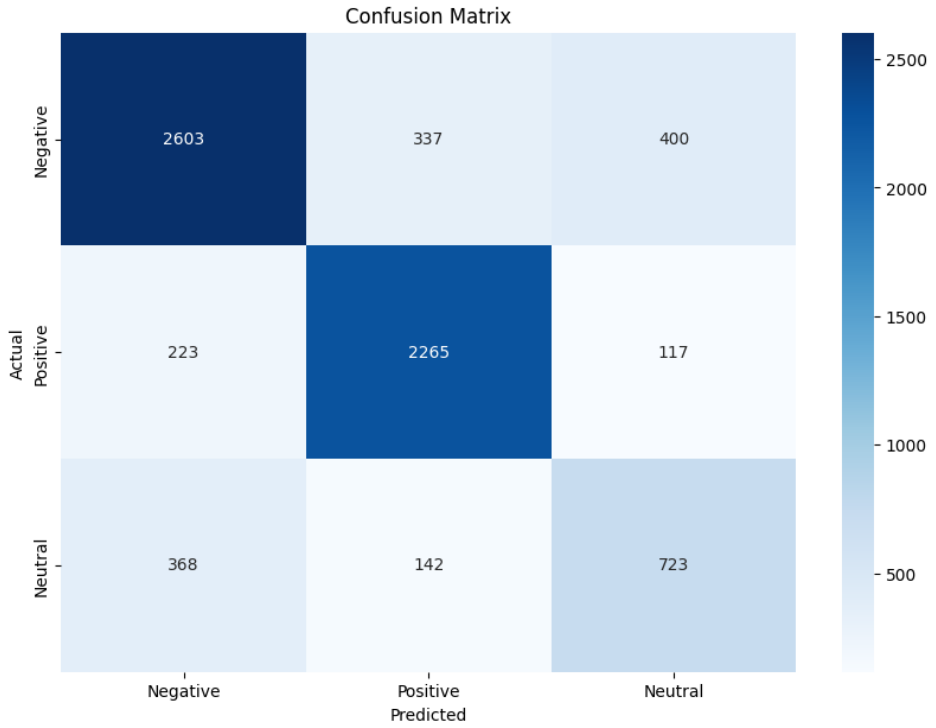


Figure 5-45-5 Confusion Matrix of ResNet50 Model

5.1.3 Hybrid Model (VGG16 + ResNet50)

The hybrid model was designed to combine the strengths of both VGG16 and ResNet50 architectures. However, the results did not meet expectations, as shown in the classification report:

	precision	recall	f1-score	support
Negative	0.67	0.67	0.67	3340
Positive	0.23	0.19	0.21	2605
Neutral	0.04	0.05	0.04	1233
accuracy			0.39	7178
macro avg	0.31	0.31	0.31	7178
weighted avg	0.40	0.39	0.40	7178

Figure 5-6 Classification Report Of Hybrid Model

The Negative category, it struggled heavily with Positive and Neutral emotions. Specifically, the model's F1-score for Positive and Neutral emotions was 0.21 and 0.04, respectively, indicating substantial difficulties in generalizing across different emotion categories.

The confusion matrix for this hybrid model confirms these struggles, with significant misclassifications across all categories. The negative impact of combining two models without sufficient synergy is evident in this case, highlighting that **deeper or combined architectures do not always guarantee improved performance. In fact, model complexity must be carefully balanced with the ability to generalize across the target dataset.**

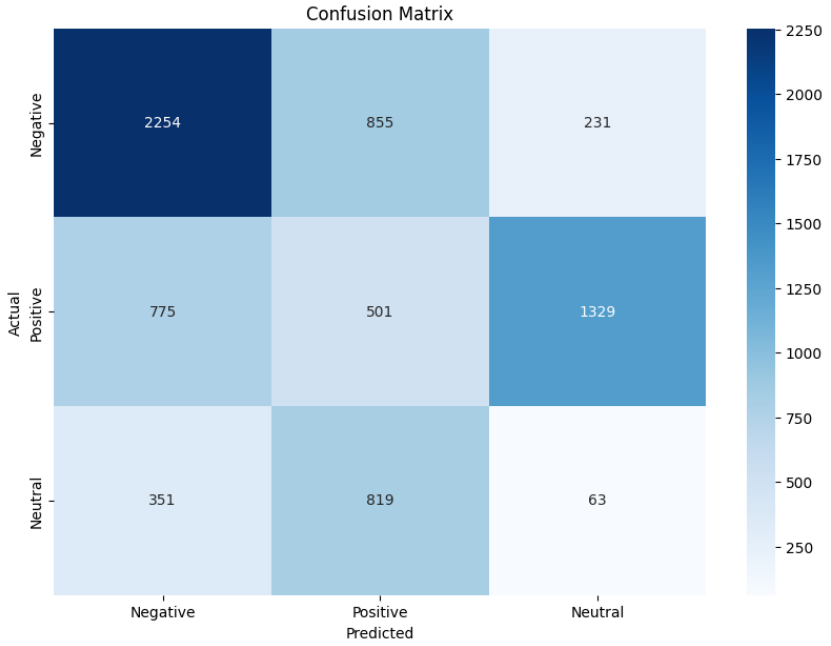


Figure 5-7 Confusion Matrix of hybrid Model

5.2 Comparative Analysis of Model Performance

Comparing the performance of the three models shows that ResNet50 achieved the highest overall accuracy and significantly outperformed both the VGG16 and hybrid models in terms of precision, recall, and F1-score across all emotion categories. The superiority of ResNet50 can be attributed to its deeper architecture, which allows for better feature extraction and a more refined understanding of complex patterns in the data.

However, the VGG16 model still performed reasonably well on Negative and Positive emotions, although it struggled with Neutral emotions. The hybrid model underperformed, demonstrating that combining architectures without sufficient fine-tuning may result in degraded performance. The poorer performance of the hybrid model also exemplifies that **deeper models or more complex models do not always yield better results**. As seen in the case of ResNet50, careful fine-tuning and proper architecture design are crucial for achieving high performance.

5.3 Discussion

The results clearly demonstrate the importance of choosing the right architecture and fine-tuning strategy when applying deep learning models to emotion recognition tasks. ResNet50 emerged as the best-performing model in this study, largely due to its ability to handle complex patterns and its residual connections, which enable the model to train deeper without the issue of vanishing gradients.

VGG16, while still a competent model, struggled with more nuanced emotional categories such as Neutral, likely due to its simpler architecture and lack of advanced techniques like residual learning. The hybrid model, which aimed to leverage the strengths of both VGG16 and ResNet50, underperformed, reinforcing the fact that **complexity alone does not guarantee better performance.**

One key takeaway is that **deeper models like ResNet50 can effectively generalize if appropriately fine-tuned**. However, merely combining models, as seen with the hybrid architecture, may result in poor performance if the integration is not carefully managed. This outcome underscores the critical need for balance between **model complexity and generalization capacity.**

5.4 Performance Comparison with State-of-the-Art Techniques on FER-2013 Dataset

5.4.1 Overview of My Best Model Performance(Resnet50)

The performance metrics of my emotion detection model using ResNet50 with transfer learning and initial training with VGU and IIMI dataset are summarized as follows:

Metrics	Values
Accuracy	78.0%
Epochs	19
Dataset of Testing	Fer-2013

This model demonstrates a high level of accuracy of **78%** on the FER-2013 dataset, achieved in a relatively short training period.

5.4.2 State-of-the-Art Performance on FER-2013 Dataset

According to Canal et al.(2022), Gan(2018) Achieved the highest accuracy on Fer-2013 dataset. The author of the study evaluated various models for emotion detection on the FER-2013 dataset using different numbers of epochs: 6, 8, 12, 20, and 25. The top average accuracy achieved was **64.24%** by **the AlexNet** model trained for 20 epochs.

Metrics	Values
Accuracy	64.24%
Epochs	20
Dataset of Testing	Fer-2013

Epochs	VGG lrate= 0.01	ResNet lrate= 0.01	Google Net lrate= 0.001	AlexNet lrate=0.001
25	0.5964	0.49	0.6391	0.6374
20	0.6098	0.4916	0.6073	0.6424
12	0.6017	0.4833	0.6171	0.5978
8	0.5836	0.4869	0.5966	0.5844
6	0.4858	0.4621	0.5594	0.5546

Figure 5-8 Screenshot of the results of accuracy models used by Gan(2018)

5.4.3 Comparative Analysis With State of The art Performance on Fer-2013 dataset

- **Accuracy**: My ResNet50 model with transfer learning achieves an accuracy of **78.0**% on the FER-2013 dataset, significantly surpassing the highest accuracy reported by **AlexNet (64.24%)** and other state-of-the-art models on Fer2013 Dataset. This indicates that your model outperforms existing methods in emotion detection.
- **Epochs**: My model achieved this high level of accuracy in just **19 epochs,** whereas Google Net reached its accuracy of 63.91% in 25 epochs. Other models, including AlexNet, VGG, and ResNet, required 20 epochs. The fewer epochs needed by my model highlight its efficiency in training while still achieving superior performance.
- **Initial Training**: To enhance generalization, my model was initially trained with the VGU and IIMI datasets. This pre-training likely contributed to the model's improved performance on the FER-2013 dataset by broadening its feature extraction capabilities and reducing overfitting.

  Note: All models, including my ResNet50 model and the models evaluated by Gan (2018), were tested on the FER-2013 dataset. This consistent dataset basis ensures a fair comparison across different architectures and training methodologies.

  My ResNet50 model with transfer learning initially trained on VGU and IIMI dataset demonstrates a notable improvement over state-of-the-art techniques for emotion detection on the FER-2013 dataset test images. Achieving an accuracy of 78.0% in fewer epochs compared to other models illustrates the effectiveness of approach. The initial training with the VGG and IIMI datasets contributed to better generalization and robustness. This performance enhancement reflects the advantages of using advanced transfer learning techniques and a robust architecture like ResNet50.

5.5 Results (User Friendly Interface)

Following the identification of the ResNet50 model with transfer learning as the best performing model, achieving an accuracy of 78.0% on the FER-2013 dataset, this model was selected for deployment to analyze video content. The detailed process of this deployment, including model integration, backend development, and user interface design, has been thoroughly discussed in the Implementation chapter.

In this section, present screenshots of the user interface and sample outputs to illustrate how the model operates in the deployed environment.

5.5.1 User Interface Overview

The user interface was designed to facilitate easy interaction with the deployed model. Users can upload videos for analysis, and the interface provides real-time feedback on the detected emotions.

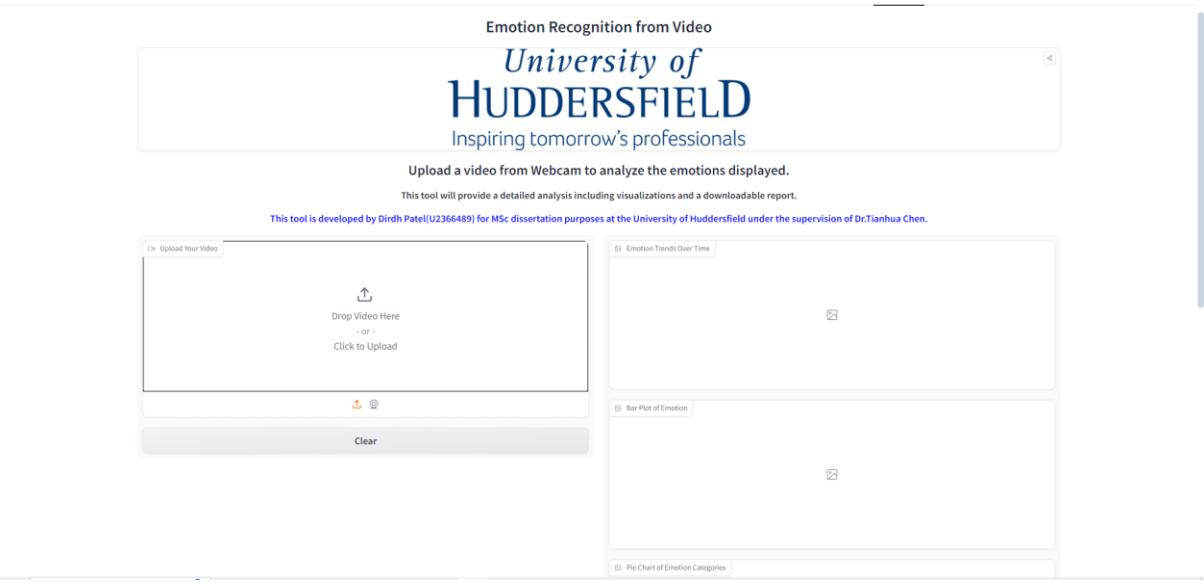


Figure 5-9Screenshot of the user interface where users can upload a video file for emotion analysis.

5.5.2 Example of Video Analysis Output

Once a video is uploaded, the backend processes it by extracting frames and running them through the deployed ResNet50 model. The predicted emotions are then displayed in a timeline format, allowing users to see how emotions fluctuate throughout the video.

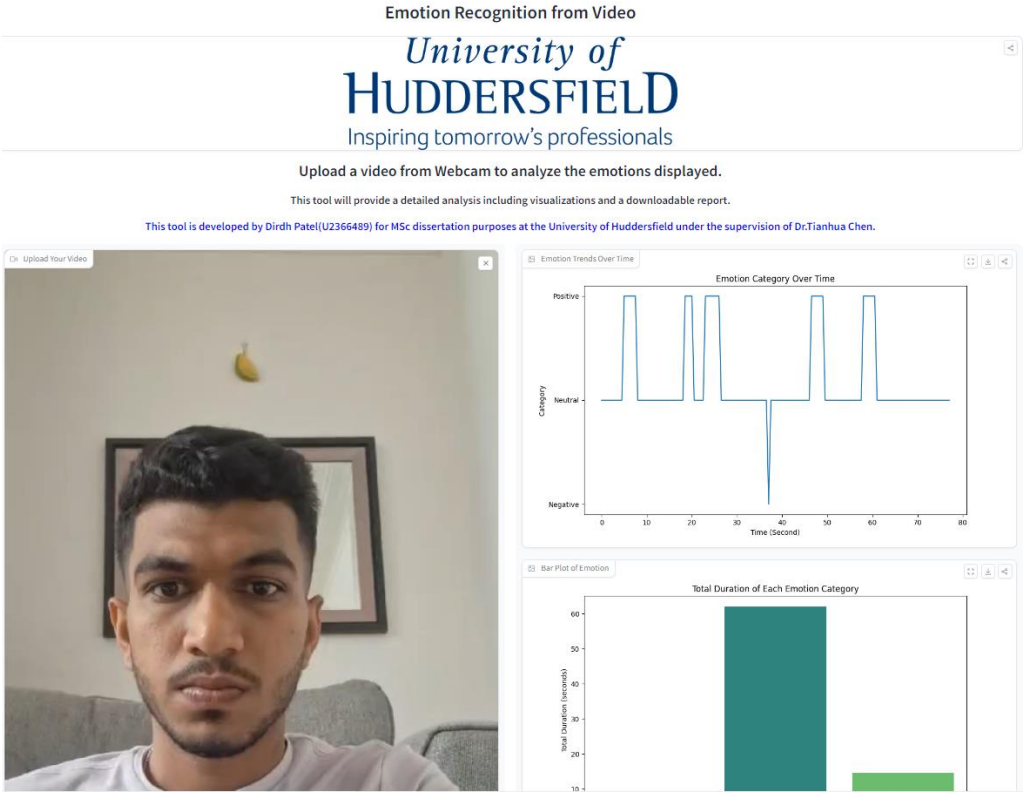


Figure 5-10 Screenshot of the output showing the detected emotions across the video part 1

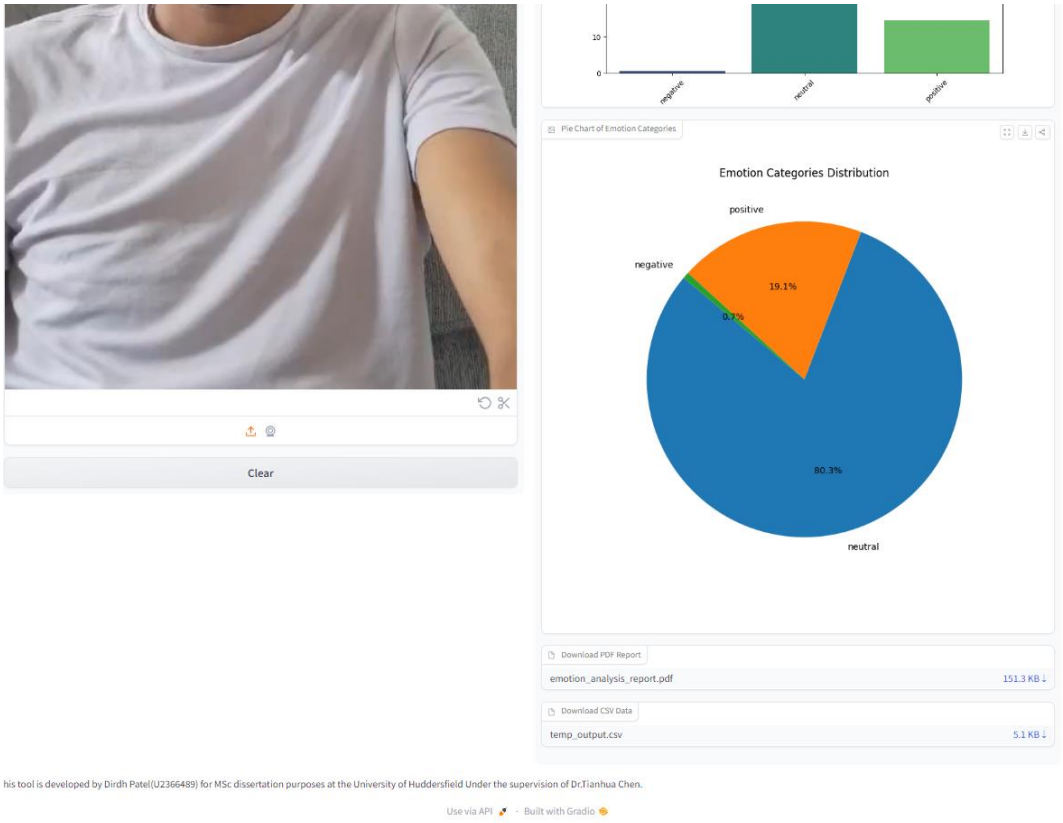


Figure 5-11 Screenshot of the output showing the detected emotions across the video part 2.

These screenshots illustrate the practical deployment of the ResNet50 model for video analysis, showcasing its ability to accurately detect and display emotions across video content. By integrating this model into a user-friendly interface, the system effectively demonstrates its potential for real-world applications, enabling users to gain valuable insights from emotional data in videos The model is deployed on the following link:

https://huggingface.co/spaces/dirdh2366489/Facialemotionreco

6 Future Works, Professional and Personal Development and Conclusion

6.1 Future Works

Building on the findings of this chapter, several avenues for future work can be explored to enhance the performance and applicability of the emotion detection system:

6.1.1 Training on Diverse Datasets:

One way to improve the generalisation power of the model would be to train it on Dataset containing individuals from as many different cultures and demographics as possible – which would help the model learn to recognise emotion more accurately among different populations, in turn curtailing the risk of bias while also improving its real-life applicability.

6.1.2 User Data Integration for Model Retraining:

A promising direction involves the development of a system that can continuously improve by retraining the model using user-specific data. Such a system would allow the model to adapt to the unique emotional expressions of individual users, providing more personalized and accurate emotion detection over time. This approach would be particularly useful in applications where long-term user interaction is expected.

6.1.3 Integration with Face Identification Models:

Another significant enhancement could be achieved by merging the emotion detection model with face identification models. This integration would enable the system to provide a comprehensive analysis of an individual's emotions within a video, offering personalized reports based on both emotional expression and identity. Such a system could be particularly valuable in security, surveillance, and user experience applications, where understanding both identity and emotion is crucial.

6.1.4 Improvement of Generalization Techniques:

Due to the overfitting during training, future research could incorporate different regularisation techniques such as data augmentation, dropout and weight decay to enhance the generalisation ability of the model. Cross-validation schemes can also be implemented to ensure that the model performs well on various subsets of the data.

Proceeding either of these directions forward will increase the robustness, personalisation and generalisation of an emotion-detection system, improving its ability to handle real-world tasks. Any improvements we gain in such verification will probably increase the reliability and accuracy of our emotion detector and help expand the kinds of everyday uses for the emotion-detection system.

6.2 Professional and Personal Development

This project has been a true learning experience, both for personal growth and as a preparation and training for my future work as an AI Developer . The technical and non-technical skills that I have learned here will be the foundation and the start of a long career in the AI field.

In terms of technical skills, I can now work with modern advanced deep learning models like VGG16, ResNet50 and other hybrid variants. With my hands-on practice I learned how to implement these models and apply transfer learning – a technique where you take a pre-trained network on a specific task and apply it to a specific problem – for example, how to improve your network's performance on a task such as recognising emotions on people's faces. I also learned data-preprocessing techniques such as how to deal with large datasets, how to deal with class imbalances (when a specific class is way more common in the data than others) and how to apply various data augmentation techniques. All these experiences have expanded my understanding of how to build scalable and efficient machine learning pipelines that are applicable to a wide variety of real-world problems.

Perhaps the most difficult – but satisfying – part was the entire system development cycle: I constantly had to debug code and solve multiple technical problems, such as model convergence (gradual improvement of a model's performance during training, without it ever improving further) and performance (the fit of the data in relation to that model). Even though all the instructions were clear and labelled, there is always a possibility that something will go wrong, so I had to learn how to be patient and persevering in the face of surprises, which does occur sometimes when training the model. What is clear is that managing limited computational resources is an art in itself. For example, when the budget in time or space is very limited, I have to make decisions about model complexity, use of algorithms, and techniques such as AUTOTUNE to make data loading faster.

In addition to the technical skills resulting from this project, I've gained critical soft skills. The management and coordination of the tight deadlines together with more involved technical work has given me the experience of learning to allocate time and complete weekly milestones. Using the system with close colleagues and peers enabled me to not only strengthen the emotion detection models for accuracy and robustness, but also to seek feedback about usability and functionality. Iteratively incorporating and testing this feedback enforced the importance of the 'energetic attention to detail'.

Strong communication abilities were another skill I acquired as a result of this project. I learnt how to convey details of a technical nature , for example, my supervisor and colleagues. This project gave me the opportunity to improve my communications with my supervisor. Adhering to my supervisor's expectations was a key factor that steered the project in the right direction., through frequent meetings and feedback sessions, I was kept on track. The project also gave me experience in writing professionally. I wrote a dissertation on the project findings and had to learn how to explain complex matters in a simple and concise way, while adhering to academic standards, which significantly enhanced my professional writing skills.

Additionally, this project has encouraged me to think about other possible AI applications, especially in the field of image generation technologies. I liked the emotion detection task so much that I became interested in how generative models (eg, GANs: Generative Adversarial Networks; and VAEs: Variational Autoencoders) could be applied to other tasks, such as generating new facial expressions or creating new datasets to improve emotion recognition systems. I also want to think about these technologies and how they can be applied to creative contexts, or in healthcare, where synthetic realistic images could be useful.

Moreover, this project helped me to learn how to work with little computational power, to work with real time, to do things in a resource-efficient way. Managing the technicalities of the project with the limited hardware resources I had was important, because it taught me how to keep things simple and effective. The most important thing I learned was how to make the best of what was available; although I was using cloud services, I had to optimise my model's memory usage to keep things lightweight, make heavy use of batch processing and regularisation, and generally strike the right balance between complexity and efficiency. This experience will help me work with larger and more complex projects in the future.

Overall, this project has given me incredible experience that has honed my technical knowledge on deep learning and AI, as well as my time management, communication, and collaboration skills. I feel more confident now that I can work on AI problems in the real world and that I can approach new challenges. I have a particular interest in image generation and AI ethics and would like to continue my work here. This project has given me the building blocks to pursue my career and do more research on topics such as bias mitigation and ethical AI.

6.3 Conclusion

In this Project, I have meticulously developed and implemented a facial emotion recognition system using deep learning methodologies, culminating in a model that has shown superior performance on the FER2013 dataset. The structured approach encompassed several stages, beginning with data collection, where I utilized the VGU,IIMI and FER2013 datasets to ensure exposure to diverse emotional expressions across varied demographic groups. Following this, the data preprocessing phase involved techniques such as data balancing through class weight adjustments and data augmentation to mitigate bias, along with normalizing and resizing all images to 224x224 pixels for model compatibility. The model selection and experimentation phase tested three distinct models, including VGG-16 and ResNet-50, through transfer learning, and a hybrid model that combined elements of both, with all models initially trained on the VGU and IIMI dataset and later retrained on the FER2013 dataset for improved performance.

The training strategies were carefully designed to enhance the models' generalization abilities, initially training on the VGU and IIMI dataset before fine-tuning with FER2013 data. Evaluation metrics like the classification report and confusion matrix revealed that the ResNet-50 model outperformed the others, achieving a remarkable 78% accuracy, surpassing state-of-the-art benchmarks on FER2013 dataset. Following this, the best-performing model was deployed using the gradio Interface , which allowed for Video analysis of emotion and Visualization of Analysis , demonstrating the model's real-world applicability. The implementation chapter detailed the logical flow and code snippets involved, while the results and analysis chapter provided a comprehensive comparison of model performance, supported by relevant screenshots, thereby illustrating the effectiveness and potential of the developed system.

7 References:

Alzubaidi, L., Zhang, J., Humaidi, A.J. et al. Review of deep learning: concepts, CNN architectures, challenges, applications, future directions. J Big Data 8, 53 (2021). https://doi.org/10.1186/s40537-021-00444-8

Andreotta, M., Nugroho, R., Hurlstone, M. J., Boschetti, F., Farrell, S., Walker, I., & Paris, C. (2019). Analyzing social media data: A mixed-methods framework combining computational and qualitative text analysis. Behavior research methods, 51, 1766-1781.https://doi.org/10.3758/s13428-019-01202-8

Barrett, L. F., Adolphs, R., Marsella, S., Martinez, A. M., & Pollak, S. D. (2019). Emotional expressions reconsidered: Challenges to inferring emotion from human facial movements. Psychological science in the public interest, 20(1), 1-68.https://doi.org/10.1177/1529100619832930

Bengio, Y., Simard, P., & Frasconi, P. (1994). Learning long-term dependencies with gradient descent is difficult. IEEE Transactions on Neural Networks, 5(2), 157-166. https://doi.org/10.1109/72.279181

Bhatia, J. K., Singh, J. P., Singh, P. K., & Chauhan, V. K. (2022, December). Emotion Detection using Facial Expressions. In 2022 11th International Conference on System Modeling & Advancement in Research Trends (SMART) (pp. 1491-1498). IEEE.

Cai, Y., Li, X., & Li, J. (2023). Emotion recognition using different sensors, emotion models, methods and datasets: A comprehensive review. Sensors, 23(5), 2455. https://doi.org/10.3390/s23052455

Calvo, R. A., & D'Mello, S. (2010). Affect detection: An interdisciplinary review of models, methods, and their applications. IEEE Transactions on affective computing, 1(1), 18-37. https://doi.org/10.1109/T-AFFC.2010.1

Canal, F. Z., Müller, T. R., Matias, J. C., Scotton, G. G., de Sa Junior, A. R., Pozzebon, E., & Sobieranski, A. C. (2022). A survey on facial emotion recognition techniques: A state-of-the-art literature review. Information Sciences, 582, 593-617. https://doi.org/10.1016/j.ins.2021.10.005

Choudhary, N., Sharma, A., Rathore, V.S., Tiwari, N. (2024). Performance Comparison of ResNet50V2 and VGG16 Models for Feature Extraction in Deep Learning. Lecture Notes in Networks and Systems, vol 812. Springer, Singapore. https://doi.org/10.1007/978-981-99-8031-4_21

Chamorro-Premuzic, T., Winsborough, D., Sherman, R. A., & Hogan, R. (2016). New talent signals: Shiny new objects or a brave new world?. Industrial and Organizational Psychology, 9(3), 621-640. https://doi.org/10.1017/iop.2016.6

Deshmukh, R. S., Jagtap, V., & Paygude, S. (2017). Facial emotion recognition system through machine learning approach. In 2017 international conference on intelligent computing and control systems (pp. 272-277). IEEE. https://doi.org/10.1109/ICCONS.2017.8250725

Gan, Y. (2018, August). Facial expression recognition using convolutional neural network. In Proceedings of the 2nd international conference on vision, image and signal processing (pp. 1-5). https://doi.org/10.1145/3271553.3271584

Garcia, J. M., Penichet, V. M., & Lozano, M. D. (2017). Emotion detection: a technology review. In Proceedings of the XVIII international conference on human computer interaction (pp. 1-8). https://doi.org/10.1145/3123818.3123852

Glorot, X., & Bengio, Y. (2010). Understanding the difficulty of training deep feedforward neural networks. In Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics (pp. 249-256). https://proceedings.mlr.press/v9/glorot10a.html

He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (pp. 770778).https://openaccess.thecvf.com/content_cvpr_2016/html/He_Deep_Residual_Learning_CVPR_2016_paper.html

Howard, A., Zhang, C., & Horvitz, E. (2017). Addressing bias in machine learning algorithms: A pilot study on emotion recognition for intelligent systems. In 2017 IEEE Workshop on Advanced Robotics and its Social Impacts (ARSO) (pp. 1-7). IEEE.https://doi.org/10.1109/ARSO.2017.8025197

Huang, G., Liu, Z., Van Der Maaten, L., & Weinberger, K. Q. (2017). Densely connected convolutional networks. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (pp. 4700-4708). https://openaccess.thecvf.com/content_cvpr_2017/papers/Huang_Densely_Connected_Convolutional_CVPR_2017_paper.pdf

Kaur, D., Uslu, S., Rittichier, K. J., & Durresi, A. (2022). Trustworthy artificial intelligence: a review. ACM computing surveys (CSUR), 55(2), 1-38. https://doi.org/10.1145/3491209

Koole, S. L. (2009). The psychology of emotion regulation: An integrative review. Cognition and emotion, 23(1), 4-41. https://doi.org/10.1080/02699930802619031

Kumari, N., & Bhatia, R. (2021). Systematic review of various feature extraction techniques for facial emotion recognition system. International Journal of Intelligent Engineering Informatics, 9(1), 59-87. https://doi.org/10.1504/IJIEI.2021.116088

Li, H., Xu, Z., Taylor, G., Studer, C., & Goldstein, T. (2018). Visualizing the loss landscape of neural nets. Advances in Neural Information Processing Systems, 31, 6389-6399.https://proceedings.neurips.cc/paper_files/paper/2018/file/a41b3bb3e6b050b6c9067c67f663b915-Paper.pdf

Li, S., Deng, W., & Du, J. (2017). Reliable crowdsourcing and deep locality-preserving learning for expression recognition in the wild. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 2852-2861). https://openaccess.thecvf.com/content_cvpr_2017/html/Li_Reliable_Crowdsourcing_and_CVPR_2017_paper.html

Liu, W., Wang, Z., Liu, X., Zeng, N., Liu, Y., & Alsaadi, F. E. (2017). A survey of deep neural network architectures and their applications. Neurocomputing, 234, 11-26. https://doi.org/10.1016/j.neucom.2016.12.038

Manalu, H. V., & Rifai, A. P. (2024). Detection of human emotions through facial expressions using hybrid convolutional neural network-recurrent neural network algorithm. Intelligent Systems with Applications, 21, 200339.https://doi.org/10.1016/j.iswa.2024.200339

Martinez-Martin, N. (2019). What are important ethical implications of using facial recognition technology in health care?. AMA journal of ethics, 21(2), E180.https://doi.org/10.1001%2Famajethics.2019.180

Mascarenhas, S., & Agarwal, M. (2021). A comparison between VGG16, VGG19 and ResNet50 architecture frameworks for Image Classification. In 2021 International conference on disruptive technologies for multi-disciplinary research and applications (CENTCON) (Vol. 1, pp. 96-99).https://doi.org/10.1109/CENTCON52345.2021.9687944

Mohammad, S. M. (2022). Ethics sheet for automatic emotion recognition and sentiment analysis. Computational Linguistics, 48(2), 239-278. https://doi.org/10.1162/coli_a_00433

Mohanta, S.R., Veer, K. Trends and challenges of image analysis in facial emotion recognition: a review. Netw Model Anal Health Inform Bioinforma 11, 35 (2022). https://doi.org/10.1007/s13721-022-00376-0

Moravčík, E., & Basterrech, S. (2022). Image-Based Facial Emotion Recognition Using Convolutional Neural Networks and Transfer Learning. In Proceedings of the Fifth International Scientific Conference "Intelligent Information Technologies for Industry"(IITI'21) (pp. 3-14). Springer International Publishing. https://doi.org/10.1007/978-3-030-87178-9_1

Nidhi, & Verma, B. (2023). From methods to datasets: a detailed study on facial emotion recognition. Applied Intelligence, 53(24), 30219-30249.https://doi.org/10.1007/s10489-023-05052-y

Ntoutsi, E., Fafalios, P., Gadiraju, U., Iosifidis, V., Nejdl, W., Vidal, M. E., ... & Staab, S. (2020). Bias in data-driven artificial intelligence systems—An introductory survey. Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery, 10(3), e1356.https://doi.org/10.1002/widm.1356

Oyedotun, O. K., Al Ismaeil, K., & Aouada, D. (2021). Training very deep neural networks: Rethinking the role of skip connections. Neurocomputing, 441, 105-117. https://doi.org/10.1016/j.neucom.2021.02.004

Sarkar, A., Behera, P.R. et al. Multi-source transfer learning for facial emotion recognition using multivariate correlation analysis. Sci Rep 13, 21004 (2023). https://doi.org/10.1038/s41598-023-48250-x

Saxena, A., Khanna, A., & Gupta, D. (2020). Emotion recognition and detection methods: A comprehensive survey. Journal of Artificial Intelligence and Systems, 2(1), 53-79. https://doi.org/10.33969/AIS.2020.21005

Scherer, D., Müller, A., & Behnke, S. (2010). Evaluation of pooling operations in convolutional architectures for object recognition. In International conference on artificial neural networks (pp. 92-101). Berlin, Heidelberg: Springer Berlin Heidelberg. https://doi.org/10.1007/978-3-642-15825-4_10

Shruti Tewari; Samyak Mehta; Narayanan Srinivasan (2023). IIMI Emotional Face Database [Dataset]. https://osf.io/f7zbv

Simonyan, K. (2014). Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:1409.1556. https://gitea.sharpe6.com/Adog64/Adversarial-Machine-Learning-Clinic/raw/commit/0ba4e489bb77a0d2642923499598e309d8751435/references/Very_Deep_Convolutional_Networks_for_Large-Scale_Image_Recognition.pdf

Srinivasan, R., & Martinez, A. M. (2018). Cross-cultural and cultural-specific production and perception of facial expressions of emotion in the wild. IEEE Transactions on Affective Computing, 12(3), 707-721. https://doi.org/10.1109/TAFFC.2018.2887267

Stark, L., & Hoey, J. (2021). The ethics of emotion in artificial intelligence systems. In Proceedings of the 2021 ACM conference on fairness, accountability, and transparency (pp. 782-793). https://doi.org/10.1145/3442188.3445939

Veit, A., Wilber, M. J., & Belongie, S. (2016). Residual networks behave like ensembles of relatively shallow networks. Advances in Neural Information Processing Systems, 29, 550-558.https://proceedings.neurips.cc/paper_files/paper/2016/file/37bc2f75bf1bcfe8450a1a41c200364c-Paper.pdf

Vietnamese German University (2022). Emotion recognition Dataset [Dataset].https://universe.roboflow.com/vietnamesegerman-university-mavjh/emotion-recognition-rjl9w

Wachter, S., & Mittelstadt, B. (2019). A right to reasonable inferences: re-thinking data protection law in the age of big data and AI. Colum. Bus. L. Rev., 494.

Wang, F., Jiang, M., Qian, C., Yang, S., Li, C., Zhang, H., ... & Tang, X. (2017). Residual attention network for image classification. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 3156-3164). https://openaccess.thecvf.com/content_cvpr_2017/html/Wang_Residual_Attention_Network_CVPR_2017_paper.html

Yang, J., Ren, P., Zhang, D., Chen, D., Wen, F., Li, H., & Hua, G. (2017). Neural aggregation network for video face recognition. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 4362-4371). https://openaccess.thecvf.com/content_cvpr_2017/papers/Yang_Neural_Aggregation_Network_CVPR_2017_paper.pdf

Zhang, K., Zhang, Z., Li, Z., & Qiao, Y. (2016). Joint face detection and alignment using multitask cascaded convolutional networks. IEEE signal processing letters, 23(10), 1499-1503. https://doi.org/10.1109/LSP.2016.2603342

Zhang, T., Gao, C., Ma, L., Lyu, M., & Kim, M. (2019). An empirical study of common challenges in developing deep learning applications. In 2019 IEEE 30th International

Symposium on Software Reliability Engineering (ISSRE) (pp. 104-115). IEEE. https://doi.org/10.1109/ISSRE.2019.00020

Zhou, N. (2024). Image recognition in depth: comparative study of CNN and Pre-trained VGG16 architecture for classification tasks. In Second International Conference on Physics, Photonics, and Optical Engineering (ICPPOE 2023) (Vol. 13075, pp. 553-559). SPIE. https://doi.org/10.1117/12.3026829

Appendix

Ethical Review Form

GUIDELINES ON COMPLETING THE ETHICAL REVIEW FORM

Questions 1, 3, 4, 6 and 7 only apply if people will be involved in your project (e.g. in providing feedback on your product, in providing input on desirable aspects of your project and/or product (e.g. through questionnaires, focus groups, etc.), etc.)

Question 2 will probably only apply if you have a real client for your project.Questions 5 and 8 should be fairly self-evident.

Before completing question 9 you need to complete a Risk Analysis & Management form, which is attached at the end of this document, to identify risks and hazards thatmay arise from your project, if any.

University of Huddersfield School of Computing and Engineering

PROJECT ETHICAL REVIEW FORM

Applicable for all research, masters and undergraduate projects

☐

Project Title:	Facial emotion detection
Student:	Dirdh prafullkumar patel
Course/Programme:	MSc. Artificial intelligence
Department:	School of Computing And Engineering
Supervisor:	Dr. Tianhua Chen
Project Start Date:	7 th May 2024

☐

ETHICAL REVIEW CHECKLIST

	☐ Yes	No
1. Are there problems with any participant's right to remain anonymous?	☐	☒

☐

2. Could a conflict of interest arise between a collaborating partner or funding source and the potential outcomes of the research, e.g. due to the need for onfidentiality? ☒
3. Will financial inducements be offered? ☒
4. Will deception of participants be necessary during the research? ☒
5. Does the research involve experimentation on any of the following?

   (i) animals? ☒

   (ii) animal tissues? ☒

   (iii)human tissues (including blood, fluid, skin, cell lines)? ☒
6. Does the research involve participants who may be particularly vulnerable, e.g.children or adults with severe learning disabilities? ☒
7. Could the research induce psychological stress or anxiety for the participantsbeyond that encountered in normal life? ☒
8. Is it likely that the research will put any of the following at risk:

   (i) living creatures? ☒

   (ii) stakeholders (disregarding health and safety, which is covered by Q9)? ☒

   (iii)the environment? ☒

   (iv)the economy? ☒

9. Having completed a health and safety risk assessment form and taken all reasonable practicable steps to minimise risk from the hazards identified, are the residual risks acceptable (Please attach a risk assessment form – available at the end of this document) ☐ ☒

STATEMENT OF ETHICAL ISSUES AND ACTIONS

If the answer to any of the questions above is yes, or there are any other ethical issues that arise thatare not covered by the checklist, then please give a summary of the ethical issues and the action thatwill be taken to address these in the box below. If you believe there to be no ethical issues, please enter "NONE".

STATEMENT BY THE STUDENT

I believe that the information I have given in this form on ethical issues is correct.

Signature: DirdhPatel Date: 1st May 2024

AFFIRMATION BY THE SUPERVISOR

I have read this Ethical Review Checklist, and I can confirm that, to the best of my understanding, the information presented by the student is correct and appropriate to allow aninformed judgement on whether further ethical approval is required.

T Chen 03.05.24

Signature: _____ Date: _____

SUPERVISOR RECOMMENDATION ON THE PROJECT'S ETHICAL STATUS

Having satisfied myself of the accuracy of the project ethical statement, I believe that theappropriate action is:

The project proceeds in its present form	X
The project proposal needs further assessment by an Ethical Review Panel. The Supervisor will pass the form to the Ethical Review Panel Leader for consideration.	

Code File Links

Note: Direct Code didn't Type Here due to sole reason; Length of Report and Some part of Code Given in Implementation Chapter and Full Code Can Find on the Following links:

A. Code for Data collection and Data standarzation Process
https://colab.research.google.com/drive/1VUiNPvkJI5E-fZynvHIXUAJ5FftZzlU5?usp=sharing
B. Code for Data preproccing to Model evulation
https://colab.research.google.com/drive/14oIDIC_rbKngtlDbwEt0kT8JhKWPLB9x?usp=sharing
C. Code for Deployment Process
https://colab.research.google.com/drive/1TzvlaCRqqEdkWn-5GSErbgxo6oPQfuqg?usp=sharing

User Manual

This user manual is designed to guide you through the features and functionalities of System application, which is available for both desktop and Android devices. The system enables you to

analyze emotions from video inputs, offering seamless integration and automated processing for a smooth user experience.

System Requirements

Before using the Video Emotion Analysis System, please ensure that your device meets the following requirements:

For Desktop:

- Operating System: Windows, macOS, or Linux
- Web Browser: Google Chrome, Mozilla Firefox, or Microsoft Edge
- Internet Connection: Stable connection for uploading videos and downloading output files

For Android:

- Android Version: Android 6.0 (Marshmallow) or later
- Storage: Sufficient space for video uploads and output file storage
- Internet Connection: Stable connection for uploading videos and downloading output files

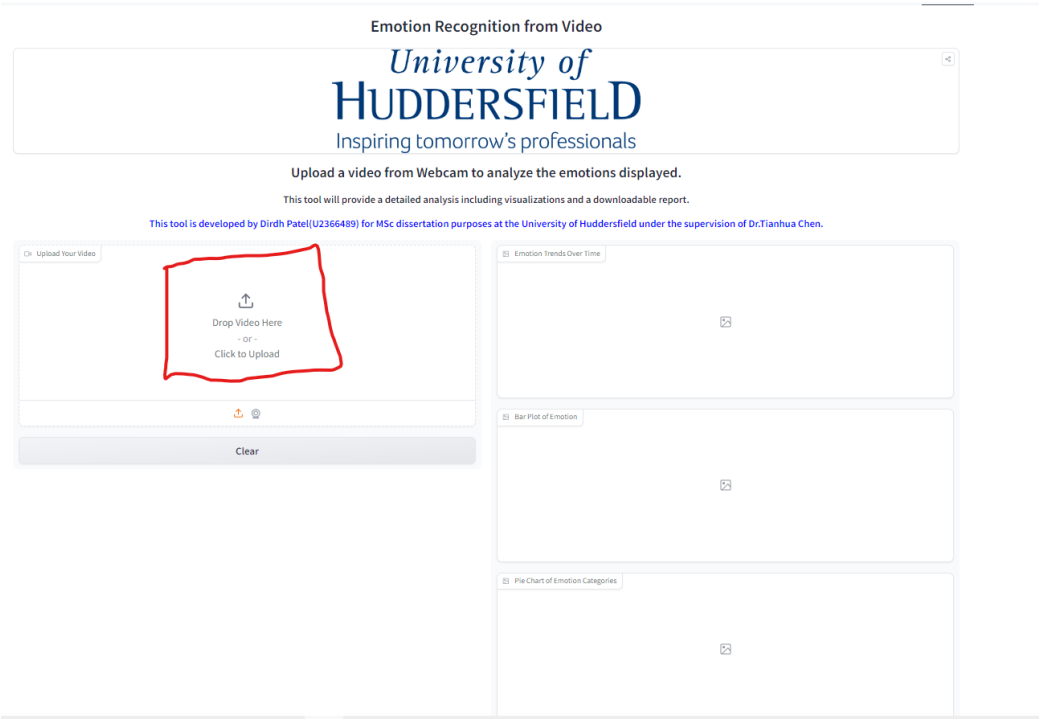
Features Overview

- **Video Upload:** Upload videos directly from your computer or Android device for emotion analysis.
- **Webcam Recording:** Use your webcam (or Android camera) to record a video and upload it directly for analysis.
- **Automated Processing:** Once a video is selected, it will automatically upload and begin processing without the need for manual submission.
- **Output Files:** The system generates four output files after processing the video:
    1. **Emotion Plot 1:** A visual representation of the detected emotions.
    2. **Emotion Plot 2:** Another visualization focusing on different aspects of the analysis.
    3. **Emotion Plot 3:** A final plot to provide additional insights.
    4. **PDF Report:** A comprehensive PDF file summarizing the analysis.
    5. **CSV Report:** A CSV file containing detailed data from the analysis.

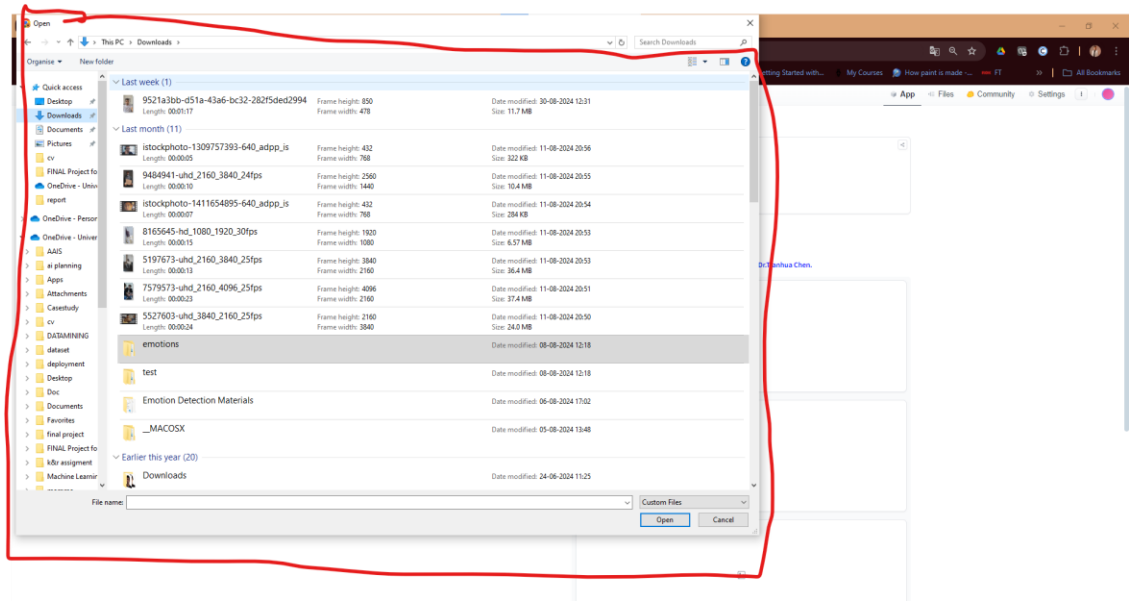
How to Use the System

Uploading a Video (Desktop and Android)

1. **Select 'Upload Video':**
    - Click the "Upload Video" button on the main dashboard.

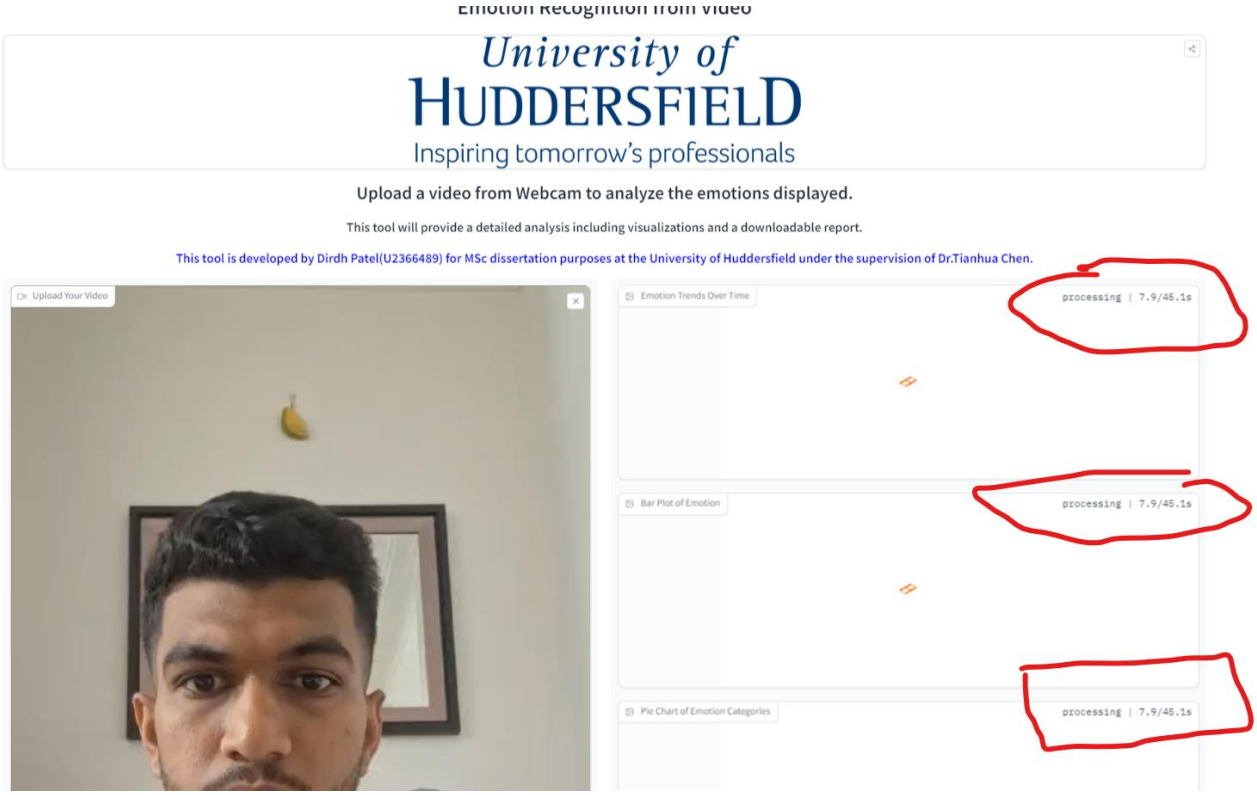


2. **Choose Video File:**
    - A file explorer window will appear. Navigate to the video file on your device and select it.



- The system supports various video formats, including MP4, AVI, and MOV.

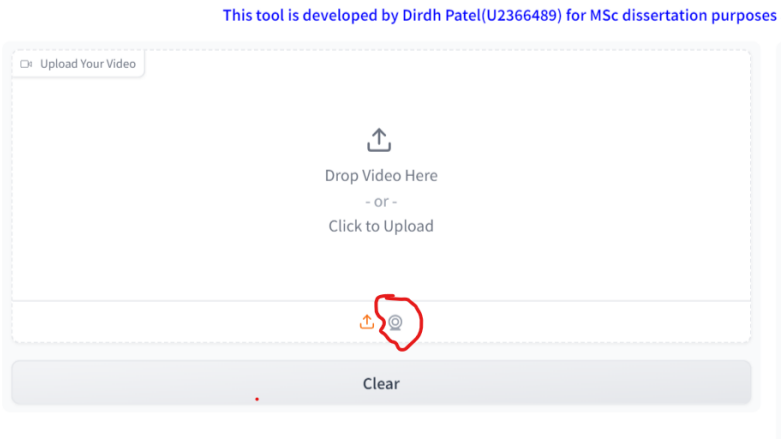
3. **Automatic Upload and Processing:**
   - Once you select a video, it will automatically begin uploading and processing.



   - No need to click a submit button—the system takes care of everything once the video is selected.

- **4.2 Recording a Video Using Webcam/Camera**

1. **Select 'Record Video':**
    - Click the "Record Video" button on the main dashboard to activate your webcam.

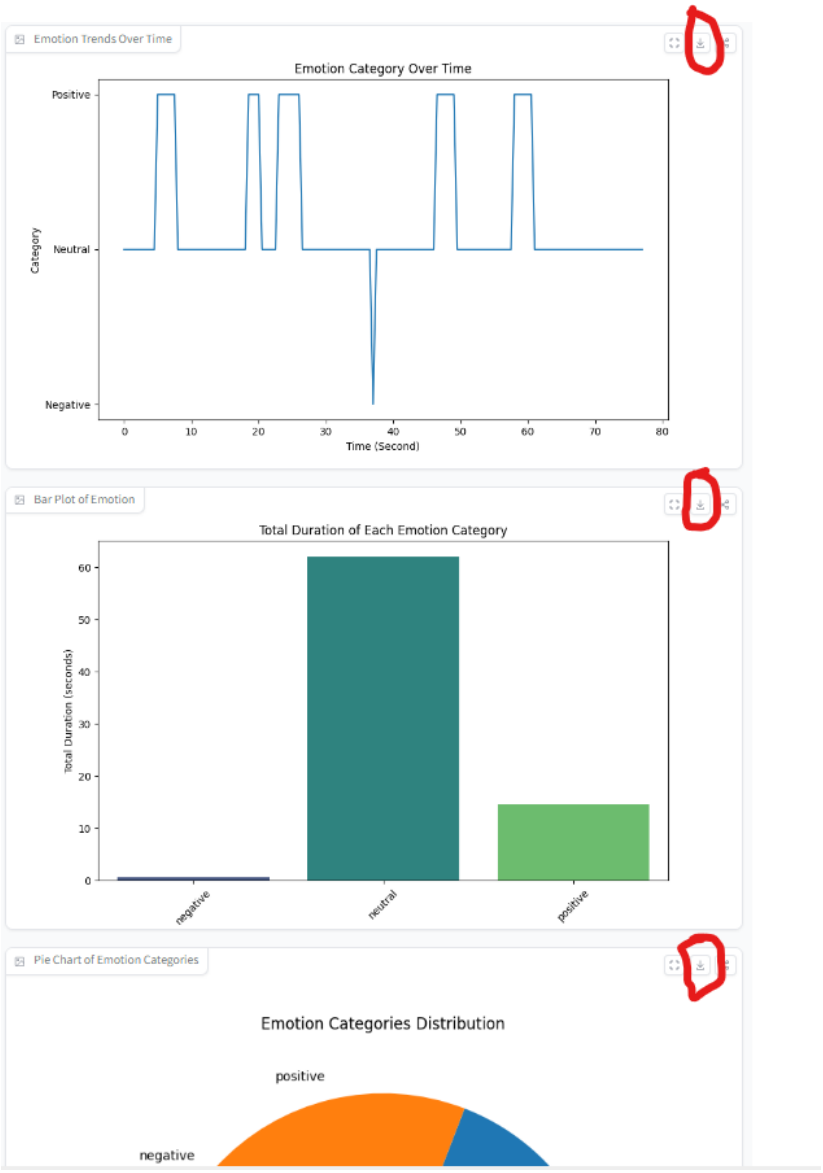


2. **Start and Stop Recording:**
    - Click or tap the "Start Recording" button to begin recording your video.
    - When finished, click or tap "Stop Recording." The video will then automatically upload for processing.

- **4.3 Downloading Output Files**

After the video has been analyzed, the system will generate four output files:

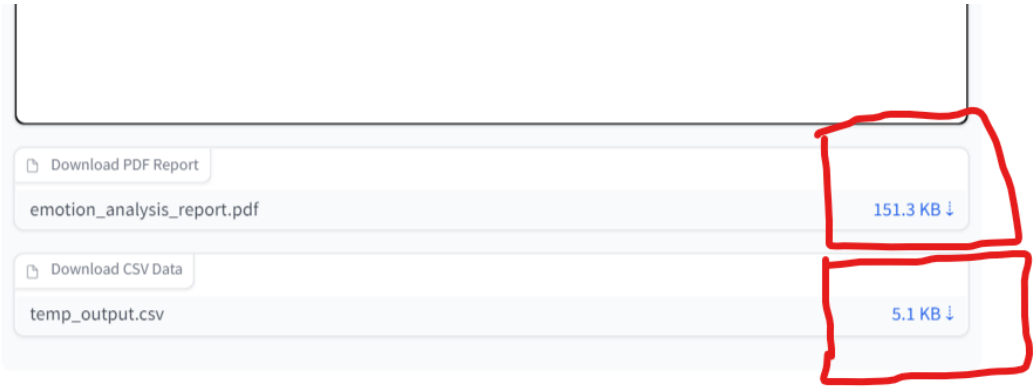
1. **Download Plots:**
    - Each of the three emotion plots will have a "Download" button.



- Click or tap the "Download" button under each plot to save the file to your device.

2. **Download PDF Report and CSV file :**

   - The PDF report summarizes the analysis in a readable format. Click or tap the "Download PDF" button to save it.
   - The CSV report contains the detailed data in a tabular format. Click or tap the "Download CSV" button to save it.

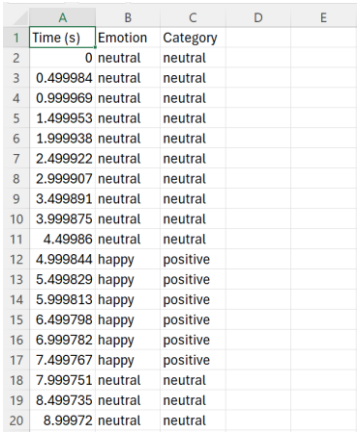


Sample Files

To help you understand the output, we have provided sample files that you can download and review:

- **Sample PDF Report:**

- **Sample CSV Report:**



	A	B	C	D	E
1	Time (s)	Emotion	Category		
2	0	neutral	neutral		
3	0.499984	neutral	neutral		
4	0.999969	neutral	neutral		
5	1.499953	neutral	neutral		
6	1.999938	neutral	neutral		
7	2.499922	neutral	neutral		
8	2.999907	neutral	neutral		
9	3.499891	neutral	neutral		
10	3.999875	neutral	neutral		
11	4.49986	neutral	neutral		
12	4.999844	happy	positive		
13	5.499829	happy	positive		
14	5.999813	happy	positive		
15	6.499798	happy	positive		
16	6.999782	happy	positive		
17	7.499767	happy	positive		
18	7.999751	neutral	neutral		
19	8.499735	neutral	neutral		
20	8.99972	neutral	neutral		

These samples provide a glimpse into what you can expect from the system after analysis.

- **5. Troubleshooting**
- **Video Upload Issues:**
    - Ensure that your video is in a supported format (MP4, AVI, MOV).
    - make sure you have a stable internet connection.
    - If the upload fails, try restarting the app or your device.
- **Webcam/Camera Not Working:**
    - Ensure that your browser or app has permission to access the webcam or camera.
    - Restart your browser, app, or device if the problem persists.
- **Download Issues:**
    - Make sure your internet connection is stable.
    - If using the Android app, check your device's storage space and internet connection.