19I620 - Innovation Practices

Facial Emotion Recognition using Deep CNN

Ariprakash R	(22I304)
Azhagudevan S	(22I306)
Nithish M	(22I337)
Saravanakumar M	(22I361)

BACHELOR OF TECHNOLOGY

BRANCH: INFORMATION TECHNOLOGY

of Anna University

APRIL 2025

DEPARTMENT OF INFORMATION TECHNOLOGY

PSG COLLEGE OF TECHNOLOGY

(Autonomous Institution)

COIMBATORE – 641 004

PSG COLLEGE OF TECHNOLOGY

(Autonomous Institution)

COIMBATORE – 641 004

Bonafide record of work done by

Ariprakash R	(22I304)
Azhagudevan S	(22I306)
Nithish M	(22I337)
Saravanakumar M	(22I361)

Dissertation submitted in partial fulfillment of the requirements for the degree of

BACHELOR OF TECHNOLOGY
BRANCH: INFORMATION TECHNOLOGY
of Anna University

APRIL 2025

……..……	……..……...
Dr. Vinoth Kumar B	**Dr.K Umamaheswari**
Faculty guide	Head of the Department

Certified that the candidate was examined in the viva-voce examination held on**…….**

……	……
(Internal Examiner)	(External Examiner)

ACKNOWLEDGEMENT

We express our sincere thanks to **Dr.K.Prakasan,** Principal, PSG College of Technology for providing an opportunity to take up the project work.

We are greatly indebted to **Dr.K.Umamaheswari,** Professor and Head, Department of Information Technology for her kind support and encouragement given to complete the project.

We profusely thank our program coordinator, **Dr. Anitha Kumari K,** Associate Professor, for constant support and encouragement for the completion of the project.

We express our sincere thanks to our tutor **Dr. Vairam T,** Assistant Professor, Department of Information Technology, for his constant support and guidance that played a vital role in completing our project.

We express our humble and profound gratitude to our project guide **Dr. Vinoth Kumar B**, Professor, Department of Information Technology for her valuable guidance and timely advice to complete the project work successfully.

We would like to extend our sincere gratitude for the invaluable support and collaboration throughout the project. Their guidance, resources, and encouragement have been instrumental in the successful completion of our work

ABSTRACT

Facial emotion recognition (FER) is a crucial task in affective computing, enabling machines to identify human emotions by analyzing facial expressions.

Our project aims to develop a cutting-edge FER model by leveraging deep learning techniques, incorporating an improved Xception-based architecture with an attention mechanism to improve accuracy.

The datasets used in this study are CK+ and FER2013, which involve preprocessing facial images through standardization and reshaping them into a 48×48 grayscale format. The architecture of the model is made up of three main parts: the entry flow, middle flow, and exit flow, which use separable convolutions to efficiently extract features. An extra layer is added to guide the model's focus towards important facial regions, enhancing its ability to learn effectively. Techniques such as rotation, shifting, and zooming are employed to improve generalization in data augmentation.

The model is trained using the Adam optimizer with categorical cross-entropy loss and evaluated on validation and test sets. Performance metrics such as accuracy, confusion matrices, and loss curves are analyzed. The model that consistently achieves the highest performance is safeguarded using modelcheckpoint, ensuring optimal results.

This project has real-world applications in human-computer interaction, healthcare, and security surveillance, offering an efficient deep learning-based system for facial recognition. By integrating the attention mechanism with Xception, the model's performance is greatly enhanced, making it a dependable solution for real-time emotion detection.

TABLE OF CONTENTS

CHAPTER **Page No.**

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1

INTRODUCTION

Facial emotion recognition is an interdisciplinary field combining computer vision and affective computing.Advances in machine learning have enabled automated analysis of subtle facial cues.Deep neural networks classify and interpret complex emotional expressions from facial images. Datasets such as AffectNet have provided diverse, in-the-wild samples for robust model training [1].The inherent challenge lies in addressing variations in lighting, pose, and occlusion.Researchers continuously refine algorithms to achieve higher recognition accuracy.

The evolution of facial emotion recognition has been driven by rapid improvements in computational methods.Innovative deep learning techniques have significantly boosted performance in real-world scenarios.Modern approaches [2] leverage convolutional neural networks to extract and interpret facial features.Large-scale datasets and crowdsourcing have advanced the accuracy of these methods. The fusion of local and global features plays a crucial role in effective recognition.Ongoing research aims to further enhance human-computer interaction through precise emotion detection.

A comprehensive survey of affect recognition methods underscores the diversity of strategies employed.Audio-visual fusion emerges as a promising approach to decipher emotional states.Facial expression analysis remains a core component [3] in understanding human affect.Traditional methods have gradually evolved into complex, multi-modal systems.These approaches tackle challenges like varying illumination and occlusions.Researchers continue exploring integrated frameworks for robust emotion detection.

Dynamic texture recognition has influenced facial expression analysis through innovative feature extraction.Techniques like Local Binary Patterns [4] capture subtle motion and texture details. Such methods have been pivotal in mitigating issues arising from illumination changes.Robust facial recognition systems depend on combining both static and dynamic features.Researchers have successfully applied these techniques to enhance emotion detection.Ongoing advancements focus on achieving real-time processing capabilities.

Integrating modality-specific deep networks has expanded the horizons of emotion recognition.Combining video data with static image analysis enriches dynamic expression understanding [5] .This strategy leverages multiple neural network architectures for robust feature

extraction.It adapts to varying emotional contexts with improved reliability and speed.Deep learning frameworks are optimized continuously for higher efficiency.Innovative network designs drive the field toward more intuitive human interfaces.

Facial emotion recognition continues to bridge the gap between artificial intelligence and human psychology.Technological progress now allows systems to interpret a wide range of subtle emotions.The integration of diverse data sources enhances the robustness of recognition models.Large, annotated datasets are critical for training accurate and resilient systems.Algorithms[1] are continually refined to capture the complexity of human expressions.This evolution is pivotal for developing interactive and assistive technologies.

The future of facial emotion recognition is poised for breakthroughs with emerging technologies.Integration with augmented reality and virtual environments promises exciting new applications.Cross-disciplinary [2] collaboration fuels the innovation of more adaptive recognition systems.These advancements are expected to yield more personalized, empathetic computing experiences.Real-time applications continue to expand the practical impact of this technology.Continuous improvement and cross-domain synergy drive the field toward new frontiers.

Advancements in attention-based convolutional neural networks have refined facial emotion recognition by emphasizing critical regions of the face.These models dynamically adapt to capture subtle variations in expression.Selective focus enables [9] robust detection even in challenging lighting conditions.The architecture mitigates interference from background noise and occlusions.Innovations in attention mechanisms are driving real-time analysis improvements.

Multi-modal integration techniques are pushing the boundaries of emotion recognition through the fusion of visual and contextual cues.Researchers are combining data from various sources to build comprehensive affective models.The synergy of deep learning with traditional processing methods results in higher accuracy.Crowdsourced and diverse datasets [2] further enhance the robustness of these approaches.Such methodologies pave the way for advanced, context-aware emotion analysis applications.

Collaborative networks leveraging multi-stream data have revolutionized dynamic expression analysis.Deep architectures [5] are employed to simultaneously learn spatial and temporal features.This dual approach significantly improves the recognition of fleeting and spontaneous emotionsThe emphasis on adaptive learning mechanisms bolsters the reliability of emotion classification. Ongoing research in this area is instrumental in shaping the future of intelligent, emotion-aware systems

1.1 Problem Statement

Advancements in attention-based convolutional neural networks have refined facial emotion recognition by emphasizing critical regions of the face.These models dynamically adapt to capture subtle variations in expression.Selective focus [9] enables robust detection even in challenging lighting conditions.The architecture mitigates interference from background noise and occlusions.Innovations in attention mechanisms are driving real-time analysis improvements.This progress highlights the potential for more empathetic human-computer interactions.

Multi-modal integration techniques are pushing the boundaries of emotion recognition through the fusion of visual and contextual cues.Researchers are combining data from various sources to build comprehensive affective models [2] .These integrated systems excel at capturing both overt expressions and nuanced emotional signals.The synergy of deep learning with traditional processing methods results in higher accuracy.Crowdsourced and diverse datasets further enhance the robustness of these approaches.Such methodologies pave the way for advanced, context-aware emotion analysis applications.

Collaborative networks leveraging multi-stream data have revolutionized dynamic expression analysis.Deep architectures are employed to simultaneously learn spatial and temporal features [5].This dual approach significantly improves the recognition of fleeting and spontaneous emotions.Fine-tuning these models allows them to perform well across varied environmental conditions.The emphasis on adaptive learning mechanisms bolsters the reliability of emotion classification.Ongoing research in this area is instrumental in shaping the future of intelligent, emotion-aware systems.

1.2 Deep Learning for Image Classification

The main objective of this project is to create an advanced facial emotion recognition (FER) system by utilizing an Xception-based deep learning model and incorporating attention mechanisms. The main objective is to enhance emotion classification accuracy by improving feature extraction and model generalization. This project will involve several stages, such as data preprocessing, augmentation, model training, evaluation, and deployment, all aimed at recognizing seven fundamental

emotions: anger, disgust, fear, happiness, sadness, surprise, and neutrality. It will utilize datasets such as CK+ and FER2013 emotion datasets to train and validate the model. Advanced data augmentation techniques will be implemented to improve model robustness against variations in lighting, pose, and occlusions.This FER system has applications in various domains, including human-computer interaction, mental health analysis, surveillance, education, and customer service. The model will be fine-tuned for real-time performance, making it suitable for integration into mobile and web applications. Furthermore, the project investigates the application of transfer learning and ensemble learning techniques to improve model performance even more.

1.3 Importance of Attention Mechanisms in CNNs

Attention mechanisms are vital in improving the efficiency of convolutional neural networks (CNN) by directing their focus towards the most significant regions of an image, while diminishing the importance of less informative areas. Traditional CNNs perform convolution operations uniformly on the entire image, treating all pixels equally, which can result in the loss of crucial details. Attention mechanisms address this limitation by dynamically allocating varying levels of importance to different regions within an image, enabling the model to focus on the most crucial parts. This enhances feature representation, resulting in improved accuracy in tasks such as object detection, facial recognition, and medical imaging.

One of the significant advantages of attention mechanisms in CNN is their capacity to improve the selection of spatial features. Spatial attention directs the model's focus to significant regions within an image, enabling it to concentrate on relevant objects while disregarding the surrounding background noise. For instance, in the field of facial emotion recognition, attention layers can focus on the eyes, mouth, and eyebrows, while disregarding less informative areas such as the forehead or chin. By selectively focusing on relevant patterns, the model can effectively extract meaningful information, minimizing the impact of irrelevant features.

Channel attention, a different kind of attention mechanism, focuses on highlighting significant feature maps while diminishing less relevant ones. Rather than giving equal importance to all channels, channel attention dynamically modifies the significance of each channel based on its relevance to the task at hand. Techniques such

as squeeze-and-excitation (SE) networks utilize channel attention to determine the significance of each feature map and adjust them accordingly. This enables CNNs to dynamically enhance crucial features, resulting in enhanced classification accuracy and resilience.

By incorporating both spatial and channel attention mechanisms, the performance of convolutional neural networks can be enhanced through a unified framework. One effective method is the convolutional block attention module (CBAM), which sequentially applies spatial and channel attention to enhance feature maps. By combining spatial and channel-wise refinements, CBAM enables convolutional neural networks to prioritize the most informative features while minimizing redundancy, resulting in substantial enhancements in image classification and object detection tasks.

Attention mechanisms also play a role in interpretability in CNNs, offering insights into how models arrive at their decisions. Visualizing attention maps aids researchers and developers in comprehending which sections of an image have the greatest impact on the model's predictions. This transparency is especially important in critical applications like medical diagnosis, where knowing the model's focus areas can help identify diseases from medical scans. Attention-based models have shown remarkable effectiveness in identifying abnormalities in X-Rays, MRIs, and CT Scans, making them indispensable in the field of modern healthcare.

One of the benefits of attention mechanisms is their capacity to enhance the efficiency of CNNs by minimizing computational costs. Rather than treating all image regions equally, attention mechanisms prioritize significant areas by allocating more computational resources, while disregarding redundant information. By selectively processing information, CNNs become more efficient and suitable for real-time applications like autonomous driving and video surveillance, as unnecessary computations are minimized.

Additionally, attention mechanisms assist CNNs in managing variations in scale, pose, and occlusions, which are frequently encountered difficulties in image classification tasks. By dynamically adapting focus based on context, attention layers enable CNNs to identify objects even when they are partially obscured, rotated, or resized. This flexibility improves the generalization ability of CNNs, allowing them to

perform effectively on various datasets and real-world situations.

Recent progress in attention mechanisms has paved the way for the creation of self-attention techniques, like vision transformers (VITS), which extend attention beyond CNNs. Unlike traditional CNNs that focus on local features, self-attention allows for global context modeling, enhancing performance in large-scale image recognition tasks. Hybrid models that combine self-attention with convolutional neural networks have demonstrated promising outcomes, leveraging the strengths of both architectures to enhance feature learning.

1.4 Role of Data Augmentation in Deep Learning

Data augmentation is a vital component in deep learning as it artificially enlarges the training dataset, enhancing model generalization, and mitigating overfitting. In practical scenarios, obtaining extensive datasets with labeled information can be difficult due to limitations in time, expenses, and privacy concerns. This is particularly advantageous in computer vision tasks such as image classification, object detection, and facial emotion recognition, where models need to identify patterns in various conditions.

One of the key advantages of data augmentation is its capacity to enhance model generalization. Deep learning models frequently rely on memorizing the training data rather than extracting meaningful features, resulting in subpar performance when presented with new samples. Augmentation introduces changes in the training data, compelling models to adjust and identify crucial patterns instead of simply memorizing specific instances. This guarantees that the model functions effectively in real-life situations with a wide range of input conditions.

In image processing, common data augmentation techniques involve rotating, flipping, cropping, scaling, adjusting brightness and contrast, and adding noise to the images. These changes mimic real-life scenarios like different camera angles, lighting conditions, and occlusions, enhancing the model's ability to handle various situations. For instance, in facial emotion recognition, flipping and rotating the image aid the model in identifying expressions regardless of slight head tilts, while brightness adjustments enhance its performance in different lighting conditions.

In addition to standard transformations, advanced augmentation techniques such as generative adversarial networks (gans), synthetic data generation, and style transfer are gaining popularity. Gans has the ability to generate lifelike synthetic images, enhancing the quality of datasets used for training. Style transfer enables models to adjust their style to various domains, enhancing their ability to perform well on different datasets. These cutting-edge methods assist in reducing data scarcity and bias in datasets by guaranteeing equal representation of classes.

Augmentation proves to be highly beneficial in addressing class imbalances, where specific categories have a significantly lower number of training samples compared to others. By expanding the dataset with more examples of underrepresented classes, augmentation prevents models from favoring dominant categories. This is crucial in tasks like medical image classification, where certain diseases may have limited labeled samples, necessitating augmentation to ensure fair and precise predictions.

Excessive augmentation can result in distorted data distributions, which may negatively impact the model's performance. For example, if facial images are rotated or distorted excessively, it can change the expression features, leading to misclassification. Consequently, augmentation strategies need to be thoughtfully planned to maintain important characteristics while introducing variety.

As deep learning continues to advance, data augmentation remains a crucial technique for enhancing model robustness and accuracy. It broadens the dataset, minimizes overfitting, and guarantees improved performance on unseen data. By skillfully utilizing augmentation techniques, researchers can create more dependable models for a wide range of applications, such as image recognition, medical diagnostics, autonomous driving, and facial emotion recognition.

CHAPTER 2
LITERATURE SURVEY

The field of facial emotion recognition has significantly evolved with the advent of large-scale datasets and improved feature extraction techniques. Research initiated with datasets like AffectNet, which provided diverse, in-the-wild facial images for robust model training [1]. Early studies primarily focused on traditional feature-based methods, including handcrafted features and statistical analysis. Survey studies consolidated various approaches, highlighting the strengths and limitations of classical techniques [3]. Techniques such as Local Binary Patterns were explored for their ability to capture dynamic texture variations in facial expressions [4]. These foundational works laid the groundwork for subsequent deep learning methods by establishing essential benchmarks and evaluation criteria.

With the introduction of deep learning, the field experienced a paradigm shift in facial emotion recognition. Convolutional Neural Networks (CNNs) became the backbone for automated feature extraction, leading to substantial improvements in recognition accuracy. Researchers have leveraged advanced architectures to handle the inherent variability and subtlety of human emotions [6]. Integrating modality-specific deep networks further enhanced the system's ability to capture nuanced emotional cues, especially in video data [5]. These approaches also addressed challenges related to lighting, occlusion, and expression intensity, setting new performance standards. The shift towards deep learning has paved the way for more scalable and adaptable emotion recognition systems.

Despite rapid advancements, several challenges continue to hinder the performance of facial emotion recognition systems. Variability in facial expressions, cultural differences, and environmental conditions remain significant hurdles. Researchers have experimented with deep locality-preserving learning and crowdsourcing techniques to improve model robustness and generalization across diverse datasets [2]. Surveys in the field emphasize the need for more comprehensive data annotation and advanced models that can handle spontaneous expressions effectively [3]. The literature highlights that while deep learning has addressed many issues, the subtlety of human emotion still

requires innovative approaches. Ongoing studies aim to bridge the gap between controlled experimental success and real-world application.

The literature underscores a trend toward hybrid approaches that combine traditional feature extraction methods with modern deep learning techniques. Early datasets and analytical methods provided the necessary framework for developing sophisticated neural architectures [1]. Subsequent work has focused on integrating multi-modal inputs to capture both spatial and temporal dynamics in facial expressions [5]. Real-time applications have been bolstered by advances in CNN architectures capable of rapid inference and adaptability to diverse conditions [6]. The collective findings of these studies reveal a promising yet challenging trajectory for future research, emphasizing the need for standardized datasets and robust evaluation metrics. Continued collaboration and innovation are critical to achieving a comprehensive understanding and implementation of emotion recognition in practical scenarios.

Mollahosseini A et al [2017] explored the development of a comprehensive dataset for facial emotion recognition by introducing AffectNet [1]. Their work provided a large-scale in-the-wild database with diverse annotations for expressions, valence, and arousal. The dataset has enabled the training of robust deep learning models to capture complex human emotions. They reported significant improvements in model performance on emotional annotation tasks while also noting challenges with variability and noise in real-world data. Their contribution has set a benchmark for subsequent studies and highlighted the necessity of extensive, diverse datasets. They emphasize that future work should focus on standardizing annotations and improving data quality.

Li S et al [2020] investigated reliable crowdsourcing methods combined with deep locality-preserving learning for expression recognition in the wild [2] . Their approach leveraged crowdsourced labels to enhance the reliability of emotion annotations. By integrating spatial locality constraints within deep neural architectures, they achieved notable gains in recognizing subtle emotional cues. However, they also identified inconsistencies in crowdsourced data as a major challenge for model generalization. Their study underscores the importance of robust label aggregation techniques to counteract these discrepancies. They advocate for larger, more refined datasets to further improve model accuracy.

Zeng Z et al [2009] conducted a comprehensive survey of affect recognition methods, incorporating audio, visual, and spontaneous expressions [3]. Their review synthesized a range of techniques from handcrafted feature extraction to sophisticated machine learning algorithms. They emphasized that understanding human emotions requires an integrated approach that accounts for both deliberate and spontaneous expressions. The survey also highlighted persistent challenges such as varying environmental conditions and inconsistent data quality. Despite these challenges, their work provided a foundational understanding and critical evaluation of existing methodologies. They call for more unified evaluation protocols to bridge the gap between theory and practical applications.

Zhao G, Pietikäinen M [2007] explored dynamic texture recognition using Local Binary Patterns as a means to analyze facial expressions. Their method focused on capturing fine-grained texture variations, which are essential for distinguishing subtle expressions. They demonstrated that this approach could effectively mitigate issues arising from changes in lighting and pose [4] . The study achieved robust real-time performance while remaining computationally efficient. However, the reliance on handcrafted features limited adaptability when faced with more diverse datasets. They suggested that combining traditional feature-based methods with emerging deep learning techniques could yield even more powerful recognition systems.

Kahou SE et al [2013] investigated the integration of modality-specific deep networks for emotion recognition in video. Their research focused on fusing multiple neural architectures to capture both spatial details and temporal dynamics of facial expressions [5]. This multimodal approach resulted in improved accuracy in dynamic environments. The study also addressed the challenges of processing sequential video data by effectively balancing static and motion-based cues. Despite the promising results, scaling the approach to larger, more diverse datasets remains a challenge. They stress the need for further optimization to enable real-time processing in practical applications.

2.1 Summarization of Existing System

Table 2.1 Summarization of Existing System

Reference	Title	Methodology	Limitation
[1]	AffectNet: Comprehensive Dataset for Emotion Recognition	Developed a large-scale, in-the-wild dataset with diverse annotations for expressions, valence, and arousal.	Variability and noise in real-world data; need for standardized annotations.
[2]	Reliable Crowdsourcing & Deep Locality-Preserving Learning	Leveraged crowdsourced labels with spatial locality constraints in deep CNNs to enhance recognition accuracy.	Inconsistencies in crowdsourced data affecting model generalization.
[3]	Survey of Affect Recognition Methods	Reviewed various emotion recognition techniques, including audio, visual, and spontaneous expression analysis.	Variability in data quality and lack of unified evaluation protocols for benchmarking methods
[4]	Dynamic Texture Recognition Using Local Binary Patterns (LBP)	Utilized Local Binary Patterns to capture fine-grained texture details in facial expressions.	Limited adaptability due to reliance on handcrafted features, making it less effective on diverse datasets.
[5]	Integration of Modality-Specific Deep Networks	Fused multiple neural architectures to capture both spatial and temporal dynamics of facial expressions.	Challenges in scaling the approach to larger datasets and achieving real-time processing efficiency.

CHAPTER 3

REQUIREMENT SPECIFICATION

For this project, tensorflow and keras are crucial for constructing and training the deep learning model for facial emotion recognition. Python will be the main programming language, guaranteeing smooth implementation and model optimization. The exception-based model will be employed for feature extraction and classification. Furthermore, gpu acceleration will expedite the training process and improve overall performance.

3.1 Project Specification

The project aims to create a facial emotion recognition system using a deep learning-based approach, employing an Xception-based model. The goal is to accurately classify human emotions using the CK+ and FER2013 datasets, which includes a wide variety of facial expressions. The system utilizes convolutional neural networks (CNNs) to extract spatial features from images and determine emotions like happiness, sadness, anger, surprise, and neutrality.

The project follows a systematic pipeline, beginning with data collection and preprocessing, followed by model training and evaluation. The dataset is divided into training, validation, and test sets to guarantee that the model performs well on data it has not encountered before. To improve performance, hyperparameter tuning is conducted, which involves making adjustments to the batch size, implementing dropout regularization, and selecting an appropriate optimizer.

To assess the model's performance, various metrics like accuracy, precision, recall, F1-score, and confusion matrices are examined. Visualization tools are incorporated to present real-time classification outcomes and facilitate comprehension of model behavior. The last step guarantees an interface that is easy to use, allowing users to either upload facial images or record real-time video frames for emotion classification.

The system is specifically designed for use in human-computer interaction, mental health monitoring, and intelligent surveillance. It provides scalability, enabling integration into mobile and cloud-based platforms for practical implementation in the real world. Future enhancements may include the integration of transfer learning, the inclusion of additional datasets, and the implementation of attention-based mechanisms to improve accuracy and resilience.

3.2 Functional Requirements

The system needs to correctly identify facial expressions and assign them to predefined emotion categories like happiness, sadness, anger, surprise, and neutrality. It should support image-based input for emotion detection, guaranteeing accurate classification. The model should prepare input data by converting images to grayscale, scaling pixel values, and implementing data augmentation methods to improve its ability to handle diverse scenarios.

The deep learning model should employ an Xception-based architecture to effectively extract spatial features and enhance classification accuracy. The system should have the capability to save trained parameters for future use, enabling deployment without the need for retraining.

To ensure a robust training pipeline, it is crucial to establish a sequence of steps that encompass dataset loading, preprocessing, model training, validation, and testing. The model should be capable of providing real-time predictions while maintaining low latency to ensure seamless user interaction. Furthermore, a user-friendly interface should be implemented to allow users to upload images and view the classification results in real-time.

3.2.1 Metrics of the Functional Requirements

The system's reliability in emotion recognition should be assessed using established metrics like precision, recall, and F1-score to guarantee accurate results. The model should aim to achieve a high level of accuracy in classifying the FER2013 dataset,

surpassing the performance of baseline models. It is crucial to minimize latency, guaranteeing that predictions are produced within a few milliseconds to ensure optimal performance.

To effectively train the model and assess its progress, loss functions like categorical cross-entropy should be employed. The system should strive to maintain a well-balanced confusion matrix, reducing misclassifications across all emotion categories. It is important to ensure that precision and recall values are in sync to prevent any imbalances in the classification of classes.

The efficiency of a computation should be assessed by considering the utilization of the graphics processing unit (gpu) and the speed of inference. The system should consistently deliver stable performance, regardless of the lighting conditions or the angle at which the face is viewed.

3.3 Non-Functional Requirements

The system must guarantee high availability, ensuring continuous operation even when faced with heavy computational demands. It should be able to handle a larger amount of data input without experiencing a significant decrease in performance. The model should be optimized for efficiency, minimizing gpu and memory usage while still achieving accurate results.

To safeguard data integrity and prevent unauthorized access, it is crucial to establish robust security measures. The system should be designed to handle minor failures without affecting its overall functionality. It is important to monitor the system to keep track of its performance and identify any unusual occurrences.

Usability is of utmost importance, necessitating a user-friendly interface that facilitates effortless interaction with the model. The application should be designed to seamlessly integrate with external systems using APIs. It is crucial to guarantee cross-platform compatibility, enabling deployment on various hardware and software setups.

The model should be able to perform well in a wide range of real-world situations, such as different lighting conditions, different angles of the face, and when parts of the face are hidden. It should be designed to be modular, enabling updates and modifications without requiring significant alterations to the main system. The software should follow established coding conventions to ensure ease of maintenance and readability.

Data privacy should be a top concern, guaranteeing adherence to regulations like GDPR. The system needs to handle real-time inputs without any noticeable delays. The model should be designed to provide transparent and understandable outputs, allowing users to comprehend how classification decisions are made.

The system should be capable of handling batch processing for extensive data analysis. It should provide customizable options, enabling users to adjust settings according to their specific requirements.

3.4 System Requirements

The system necessitates a powerful computing setup with a dedicated graphics processing unit (GPU) to efficiently train and run deep learning models. To ensure optimal performance, it is crucial to have a multi-core processor with a clock speed of at least 3.0 GHZ. To ensure optimal performance, the ram should have a minimum capacity of 16GB to handle large datasets and prevent memory bottlenecks during processing.

To optimize storage, it is advisable to use an SSD with a minimum capacity of 512GB, as it facilitates quicker data access and retrieval. The operating system should be either linux (ubuntu 20.04 or later) or windows 10/11, capable of supporting deep learning frameworks and essential dependencies. To use Python (3.8 or later) for this project, it is necessary to have the installation of Python, along with essential libraries like tensorflow, keras, numpy, pandas, and matplotlib.

To ensure accurate visualization and debugging of model outputs, a display with a resolution of 1920×1080 or higher is necessary. The system should be equipped with a reliable internet connection to download pre-trained models, datasets, and software

updates. Compatibility with external devices, like webcams and sensors, should be verified for practical testing.

To facilitate development, it is recommended to use an integrated development environment (IDE) such as pycharm, jupyter notebook, or vs code. Docker assistance may be required for containerized deployments. To ensure smooth collaboration and efficient project management, it is essential to install version control tools like git.

To avoid any disruptions during training sessions, it is crucial to ensure a stable power supply, as these sessions can last for hours or even days. The system must have the ability to handle multiple tasks at once, which will enhance its overall efficiency. To avoid overheating, it is crucial to have effective cooling systems, such as advanced fans or liquid cooling, in place for extended periods of usage.

3.5 Dataset Specification

The extended cohn-kanade (CK+) dataset is one of the most extensively utilized resources in facial emotion recognition research, offering a rich collection of image sequences that capture the evolution of facial expressions from a neutral state to a peak emotional display. Originally developed to enhance the capabilities of its predecessor, the CK+ dataset has been meticulously curated under controlled conditions, ensuring high-quality, consistent data that provides researchers with valuable insights into the nuances of human emotion. Each sequence in the dataset is annotated with labels corresponding to basic emotions such as anger, contempt, disgust, fear, happiness, sadness, and surprise, making it a reliable benchmark for developing and testing emotion recognition algorithms. Although the dataset was collected in controlled environments with uniform lighting and minimal occlusions, which helps in reducing noise and enhancing the clarity of the expressions, these very conditions also present limitations when it comes to generalizing the findings to real-world scenarios where factors like varying illumination, head poses, and occlusions are common. Despite these challenges, the CK+ dataset remains a cornerstone in the field, as its rigorously annotated sequences and comprehensive labeling have paved the way for the development of sophisticated deep learning models that can interpret facial cues with high precision. Researchers have

leveraged the dataset to experiment with a variety of machine learning and deep learning techniques, ranging from traditional classifiers to complex convolutional neural networks (CNNs) and more recently, attention-based models that can focus on key facial regions to improve emotion detection accuracy. Its significance in pushing the boundaries of facial emotion recognition is highlighted by its frequent utilization in comparative studies, where new algorithms are evaluated against established methods using CK+ as a benchmark. While modern datasets are increasingly addressing the diversity and variability inherent in everyday environments, the CK+ dataset continues to serve as an invaluable tool for academic research and proof-of-concept studies, particularly in the early stages of model development. Its widespread adoption in the research community has led to a deeper understanding of the challenges and complexities associated with fer, driving innovations in data augmentation, transfer learning, and domain adaptation techniques that seek to overcome the limitations of controlled datasets. In summary, the CK+ dataset represents a fundamental asset in the journey toward more robust and generalizable facial emotion recognition systems, providing both the historical foundation and the detailed annotations necessary for pushing the boundaries of how machines understand and interpret human emotions.

The FER2013 dataset is a popular and extensively utilized collection of facial images that are employed in emotion recognition tasks. It comprises around 35,887 images, each resized to 48x48 pixels in grayscale, and features seven emotion labels: anger, disgust, fear, happiness, sadness, surprise, and neutrality. The dataset is somewhat imbalanced, with the "happy" emotion being the most common, comprising nearly 9,000 images, while "disgust" has the least, with about 547 images. FER2013 is an excellent platform for deep learning techniques because of its extensive size and wide variety of emotions, making it a popular choice for research and competitions.

The dataset presents various difficulties, such as incorrect labels and inconsistent lighting conditions, which can impact the accuracy of the model. Despite the obstacles, fer2013 has been employed to attain exceptional accuracy in facial emotion recognition tasks. The dataset's wide range of images, which are not posed or manipulated, makes it ideal for practical applications like social robots and video games that adapt the level of difficulty based on the player's emotions.

CHAPTER 4

PROPOSED METHODOLOGY

4.1 Introduction

The proposed approach aims to create a highly effective facial emotion recognition system through the use of deep learning techniques. The initial step involves preprocessing the data, where pixel values are standardized and transformed into appropriate shapes for training the model. The dataset is divided into training, validation, and testing subsets to guarantee reliable model assessment.

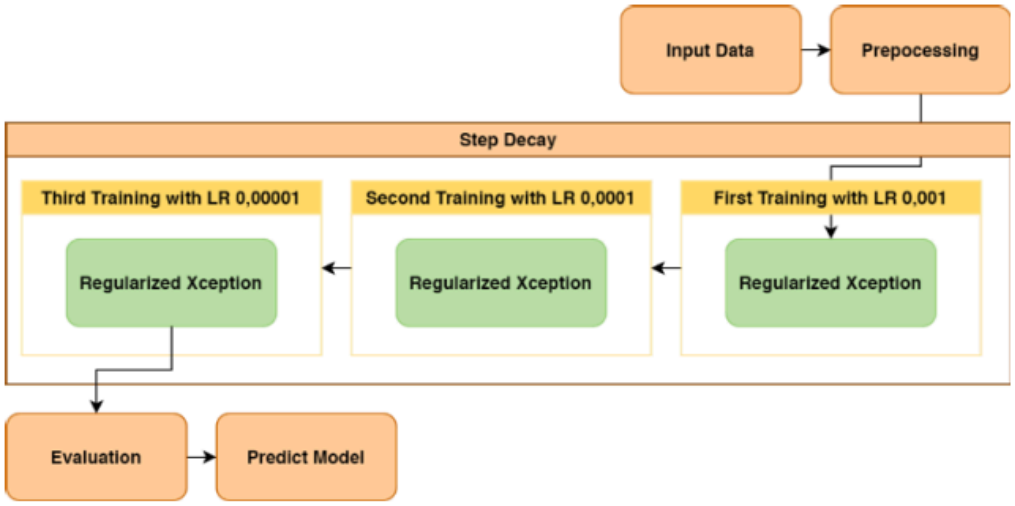

Fig 4.1 Block diagram of proposed methodology

Fig 4.1 depicts the model architecture overview.The primary architecture of the model is a convolutional neural network (CNN), which utilizes feature extraction layers to identify crucial facial patterns. Techniques like attention mechanisms and transfer learning can be incorporated to improve performance. The model is fine-tuned by adjusting hyperparameters such as learning rate, batch size, and regularization techniques to enhance its learning capabilities. The training phase entails feeding the preprocessed images into the network, with the goal of minimizing loss through backpropagation and optimization algorithms. Evaluation metrics like accuracy, precision, recall, and F1-score are employed to gauge the model's efficiency. After the training phase, the model is evaluated on data it has not seen before to assess its ability to generalize to new situations.

4.2 Model Architecture

The model architecture in this project represents an advanced convolutional neural network meticulously designed to address the challenges of recognizing subtle facial expressions from grayscale images of size 48×48, and it is built upon a comprehensive pipeline that begins with rigorous data preprocessing and augmentation, proceeds through a series of sophisticated convolutional blocks with residual connections, and culminates in a finely tuned classification head. Initially, the dataset is loaded from a specified directory where each image is read in grayscale and resized to 48×48 pixels, then normalized so that all pixel intensities fall within the [0, 1] range, ensuring consistency across inputs, simultaneously, the labels are converted into one-hot encoded vectors corresponding to seven distinct classes, which is essential for multi-class classification. The data is then stratified and split into training and testing sets to preserve the class distribution, and an imagedatagenerator is configured to apply a range of random augmentations—such as rotations up to 30 degrees, width and height shifts, zoom transformations, and horizontal flips—thus artificially increasing the diversity of the training dataset and helping the model generalize better by simulating the variability found in real-world scenarios. The network itself is ingeniously organized into three main flows: the entry flow, the middle flow, and the exit flow, each crafted to progressively transform the raw input into highly abstract representations. In the entry flow, the architecture starts with two convolutional layers, the first employs 32 filters with a stride of 2 and 'same' padding, followed by batch normalization and a relu activation, which serve to stabilize the learning process and introduce non-linearity.

This is immediately followed by a second convolutional layer with 64 filters that repeats the normalization and activation sequence, laying a robust foundation for subsequent feature extraction. A pivotal innovation in this phase is the use of residual connections: after the initial convolutions, the network enters a loop that processes the input through blocks where the filter sizes successively increase to 128, 256, and 728. Within each block, two separable convolution operations are performed—each coupled with batch normalization and relu activation—allowing the model to efficiently separate spatial and channel-wise information while significantly reducing the number of parameters compared to standard convolutions. A max pooling layer with a 3×3 kernel and a stride of 2 follows these operations, which not only reduces the spatial dimensions but also helps in focusing on the most salient features, in parallel, a shortcut connection is established by applying a 1×1 convolution to the input (adjusted to match the current block's dimensions) before combining it with the output of the main branch via

element-wise addition. This careful orchestration of convolutions and residual additions ensures that critical information is preserved across layers and that gradients propagate smoothly during training, thereby mitigating the risk of vanishing gradients. Transitioning into the middle flow, the network further deepens its feature extraction process through a series of eight repetitive blocks, each designed to refine the representations learned in the entry flow. Every block in this stage consistently employs three sequential layers of separable convolutions with 728 filters, interleaved with batch normalization and relu activations, which together create a powerful mechanism for learning complex, high-level features.

A dropout layer is integrated within each block to randomly deactivate a portion of the neurons during training, a strategy that significantly reduces overfitting by preventing the model from becoming overly dependent on any single pathway. Once again, residual connections are used to add the block's input directly to its output, ensuring that the learning process benefits from both the new transformations and the original signal. This repetitive refinement process is crucial for capturing the subtle nuances in facial expressions, as it allows the network to build a hierarchy of features that range from basic edge detections in the early layers to sophisticated patterns representing emotional cues in the later stages. The exit flow of the network is tasked with consolidating these rich features and preparing them for final classification. In this stage, the architecture begins by further processing the feature maps with separable convolutions that increment the number of filters from 728 to 1024, followed by batch normalization and relu activations, thereby deepening the network's capacity to capture intricate patterns. A max pooling operation is again applied to reduce the spatial dimensions, and a corresponding residual path uses a 1×1 convolution to ensure that the dimensions match for the subsequent addition.

The network then increases its complexity further by passing the features through additional separable convolution layers that boost the filter count first to 1536 and then to 2048, capturing an expansive range of abstract features. To bridge the gap between the convolutional layers and the final classification, a global average pooling layer is utilized, this layer computes the average of each feature map, effectively summarizing the spatial information into a single vector per channel and providing a natural form of regularization by reducing overfitting. The condensed features are finally fed into a dense layer equipped with a softmax activation function, which transforms the feature vector into a probability distribution over the seven classes, ensuring that the outputs are interpretable as class probabilities, an l2 regularization term is also applied to this dense layer to penalize large weights and further guard against overfitting. The training

process for this model is as methodically crafted as its architecture, being divided into three sequential blocks of 100 epochs each with a gradually decreasing learning rate—starting at 0.001, then reducing to 0.0001, and finally to 0.00001—thereby allowing the model to initially make large, exploratory updates to the weights and later refine its parameters with smaller, more precise adjustments. The adam optimizer is chosen for its adaptive learning rate properties, which facilitate efficient and effective training across the different phases of learning. At the end of each training block, the model's performance is evaluated on the test set using detailed metrics such as confusion matrices and classification reports, providing critical insights into the strengths and weaknesses of the network's predictions. Moreover, a checkpoint mechanism continuously monitors the validation accuracy during training and saves the best-performing model weights, ensuring that the most optimal version of the network is preserved for final evaluation.

The loss and accuracy metrics from each training phase are aggregated and visualized in combined plots that illustrate the network's learning trajectory over a total of 300 epochs, offering both a quantitative and visual narrative of the model's performance improvements over time. In sum, this model architecture is a masterful blend of modern convolutional techniques and pragmatic engineering strategies that leverage depthwise separable convolutions, residual connections, and adaptive learning schedules to form a robust system for facial expression recognition. Every stage of the network—from the meticulous data preprocessing and augmentation through the carefully layered entry, middle, and exit flows, to the final global average pooling and dense classification—has been designed to maximize the extraction of relevant features while minimizing overfitting and computational overhead. This approach not only harnesses the power of deep learning to recognize subtle variations in facial expressions but also demonstrates a sophisticated understanding of modern neural network design principles, including the importance of maintaining information flow via residual connections and the efficiency gains achieved through separable convolutions. Furthermore, the multi-stage training process, characterized by a strategic decay in learning rate and continuous performance monitoring, underscores the model's commitment to achieving high accuracy and robustness in real-world applications.

The integration of dropout and l2 regularization adds additional layers of protection against overfitting, ensuring that the network remains flexible and generalizes well to unseen data. By meticulously combining these diverse yet complementary techniques, the architecture not only excels in capturing complex visual patterns but also provides a scalable framework that can be

adapted to other challenging image classification tasks. In essence, the model stands as a comprehensive solution that effectively bridges the gap between theoretical advancements in deep learning and practical applications in computer vision, making it an exemplary candidate for tasks that demand both precision and adaptability in the analysis of subtle visual cues.

4.3 Attention Mechanism Integration

This model employs a spatial attention mechanism, which enables the network to concentrate on the most significant regions of an image while disregarding irrelevant areas. It accomplishes this by utilizing a 1×1 convolutional layer followed by a sigmoid activation function. This attention mechanism creates an attention map, which assigns greater importance to significant regions and lesser importance to less relevant areas. The attention map is multiplied with the input feature maps, emphasizing important features and diminishing irrelevant details.

In facial expression recognition, certain regions of the face, such as the eyes, mouth, and eyebrows, play a more significant role in differentiating emotions. The spatial attention mechanism dynamically emphasizes these regions, guaranteeing that the model remains focused on the most relevant parts of the image. This enhances the model's capability to identify even the slightest changes in facial expressions.

The attention layer is positioned between the middle flow and the exit flow, serving to refine the feature maps prior to the final classification. By positioning the attention module in this location, the model improves its understanding of spatial information acquired through convolutional operations. The attention map is created by analyzing the feature maps, allowing it to adapt to the specific input image rather than relying on predetermined regions.

This process operates in the manner described below:

A 1×1 convolutional layer processes the input feature maps to produce a single-channel attention map.

- The sigmoid activation function adjusts the attention values to fall within the range of 0 to 1.
- The initial maps are multiplied by the attention map, amplifying significant spatial locations.

- The enhanced feature maps are then forwarded to deeper layers for additional refinement.

By incorporating this spatial attention mechanism, the model enhances its performance by automatically learning to concentrate on important regions. This technique aids in capturing the subtle facial variations that contribute to the expression of various emotions. Unlike channel attention, which concentrates on selecting specific features, spatial attention improves feature localization, making it highly beneficial for image classification tasks such as recognizing facial expressions.

The model also gains advantages in generalization since it dynamically adjusts its focus based on various input images. Furthermore, it aids in minimizing the impact of noise by filtering out unnecessary background information, enabling the network to focus on the most relevant features of the face.

By integrating this attention mechanism, the model achieves higher accuracy in classification, improved feature extraction, and increased robustness to variations in facial expressions.

By integrating this attention mechanism, the model achieves higher accuracy in classification, improved feature extraction, and increased robustness to variations in facial expressions.

4.4 Data Preprocessing and augmentation

Data preprocessing and augmentation are crucial steps in the preparation and enhancement of deep learning models, particularly in facial expression recognition tasks. The dataset usually comprises images, with pixel values stored in a comma-separated values (CSV) format. These images are first transformed into numpy arrays, where each pixel value is divided and adjusted to a standard range. The normalization step entails dividing each pixel value by 255.0 to scale the values between 0 and 1, facilitating the neural network's learning process. Following normalization, the pixel values are transformed into a 3-dimensional array with dimensions (48, 48, 1) to align with the input specifications of convolutional neural networks (CNNs).

The dataset is divided into three main subsets: training, validation, and test data. This

division is crucial for guaranteeing that the model is trained on a specific dataset, validated on a different dataset, and tested on an additional dataset to assess its ability to generalize. Each of the emotion labels associated with the images is converted into binary vectors, representing categorical values as one-hot encoded vectors. This encoding technique enables the model to comprehend the target labels in a manner suitable for classification tasks, particularly for the seven different emotion classes involved.

To improve the model's ability to generalize and avoid overfitting, data augmentation is performed using the imagedatagenerator from keras. This tool enables real-time image modifications during the training phase, guaranteeing that the model is exposed to a wider range of images. Rotation of images up to 30 degrees is a data augmentation technique that enhances the model's ability to handle minor differences in head orientation. The width and height shift augmentation, with a 20% range, assists the model in handling translations of faces within the image, simulating misalignments that may arise in real-world situations.

Zooming is a method used to enhance the images, with a zoom range of 20%, enabling the model to learn from faces of various sizes. Horizontal flipping is employed to prevent the model from favoring one side of the face, thereby enhancing its capacity to identify symmetrical facial expressions. Furthermore, nearest-neighbor interpolation is employed to fill in any gaps in the image when applying these transformations, guaranteeing that the image structure remains intact and no distortion takes place.

By integrating preprocessing and augmentation techniques, the model is exposed to a wider range of images during training, preventing overfitting and enhancing the model's capability to generalize to new, unseen data. This is particularly crucial in facial expression recognition, where real-world conditions can differ greatly, including variations in facial orientation, lighting, and expression intensity. By implementing these preprocessing steps and augmentation techniques, the dataset becomes more diverse and resilient, enabling the model to learn and adapt to various forms of facial expressions. This process is crucial in attaining greater accuracy and guaranteeing the model's effectiveness in recognizing facial expressions in a wide range of realistic and diverse environments. Consequently, the model can identify a wide range of emotions, including happiness, sadness, surprise, anger, disgust, fear, and neutrality, with higher precision, thereby enhancing its reliability in practical scenarios.

CHAPTER 5

EXPERIMENTAL RESULTS

5.1 Implementation

The facial expression recognition model is executed on kaggle, leveraging its robust dual NVIDIA T4 GPU to expedite deep learning computations. The presence of powerful graphics processing units (GPUs) enables the smooth training of intricate Xception-based architectures that incorporate attention mechanisms. The kaggle environment is set up with all the essential libraries, such as tensorflow, keras, pandas, and numpy, guaranteeing a smooth workflow.

The model is constructed using TensorFlow and Keras, incorporating convolutional layers, separable convolutional layers, batch normalization, and an attention mechanism to improve feature extraction. Gpu acceleration is automatically activated for matrix multiplications, convolutions, and gradient computations, resulting in a substantial decrease in training time.

Tensorflow's GPU-optimized functions are utilized for batch normalization, ensuring stable gradient updates during the training process. By incorporating real-time data augmentation through imagedatagenerator, random rotations, zooming, width and height shifts, and horizontal flips are introduced, enhancing the model's ability to generalize. Augmented images are created in real-time, preventing overfitting and maintaining diversity in the dataset without consuming excessive memory.

The training process can greatly benefit from gpu-optimized optimizers like adam, which effectively update weights using adaptive learning rates. To optimize GPU memory usage while ensuring stability, a batch size of 64 is employed. The model undergoes training for 100 iterations initially, followed by adjustments to the learning rate to optimize its performance. Modelcheckpoint guarantees that only the most accurate model is saved, based on validation accuracy. The advanced storage and computational capabilities of kaggle's T4 GPU enable quick and effective training of deep cnn architectures with attention mechanisms, resulting in high accuracy with minimal training time.

5.2 Model Training and Validation

The training and validation of the Facial Expression Recognition Model are carried out using TensorFlow and Keras on Kaggle's dual NVIDIA T4 GPUs, ensuring accelerated deep learning computations. The dataset is preprocessed and split into Training, Validation, and Testing subsets, with images normalized and reshaped into (48, 48, 1) to match the input format required by convolutional layers. The labels are converted into a categorical format using to_categorical(), enabling multi-class classification.

The model architecture is based on Xception, a deep convolutional neural network that utilizes depthwise separable convolutions to improve computational efficiency. The network consists of entry flow, middle flow, and exit flow, with the addition of a spatial attention mechanism to enhance feature extraction. The training process begins with an initial learning rate of 0.001, optimized using the Adam optimizer, which adjusts the learning rate dynamically. The loss function used is categorical cross-entropy, suitable for multi-class classification problems.

Data augmentation is applied using ImageDataGenerator, incorporating random rotations, zooming, width and height shifts, and horizontal flips to introduce variability and prevent overfitting. The augmented images are generated dynamically during training, ensuring that each epoch sees slightly different variations of the same images. The model is trained using a batch size of 64, balancing computational efficiency and GPU memory utilization.

The training process is monitored using ModelCheckpoint, which saves the best-performing model based on validation accuracy. Each epoch consists of forward and backward propagation, where gradients are computed using automatic differentiation, and weights are updated accordingly. The validation dataset is used to assess the model's generalization ability, ensuring that it does not overfit the training data. After the initial training phase, the learning rate is reduced to 0.0001 and later to 0.00001, allowing for fine-tuning of the model to improve accuracy.

The entire training and validation process is executed efficiently on Kaggle's high-speed infrastructure, leveraging GPU acceleration for tensor operations, convolutions, and gradient updates. The validation accuracy is continuously monitored, and hyperparameters are adjusted to achieve optimal performance. The final trained model is saved for further evaluation and testing, ensuring its robustness for real-world facial expression recognition applications.

5.3 Performance Evaluation

The effectiveness of the facial expression recognition model is assessed using various metrics, guaranteeing a thorough evaluation of its accuracy in identifying emotions. The evaluation phase entails assessing the trained model's performance on an independent test dataset, which is essential for gauging its practicality in real-world scenarios. The test dataset comprises images that were not utilized during the training or validation phases, enabling an unbiased evaluation of the model's ability to generalize to unseen data.

The main metric used for evaluation is classification accuracy, which calculates the percentage of correctly classified images out of the total test samples. Alongside accuracy, precision, recall, and F1-score, additional metrics are calculated for each class, offering a more comprehensive understanding of the model's performance in correctly identifying facial expressions while minimizing false positives and false negatives. The categorical cross-entropy loss is evaluated to guarantee that the model does not become too specialized or too generalized.

To gain a deeper understanding of the model's performance, the learning curves, which depict accuracy and loss trends over epochs, are plotted. These curves assist in recognizing the possibility of overfitting, underfitting, or the necessity for further adjustments. The evaluation process also involves analyzing confusion matrices, which visually depict the misclassifications across various emotion classes, shedding light on the model's strengths and weaknesses in different areas.

Inference time is a crucial aspect of performance evaluation, as it measures the time required for the model to classify a single image. This is especially crucial for applications that demand immediate responses, as speed is of the essence. The model's reliability is evaluated by subjecting it to variations in lighting, occlusions, and different facial angles, guaranteeing its effectiveness in various scenarios.

In summary, performance evaluation is carried out in a systematic manner, utilizing both quantitative metrics and qualitative analysis, to guarantee that the model is not only precise but also efficient and scalable for practical facial expression recognition applications. The findings from this assessment form the basis for future enhancements and potential modifications in subsequent versions of the model.

5.4 Accuracy and Loss Analysis

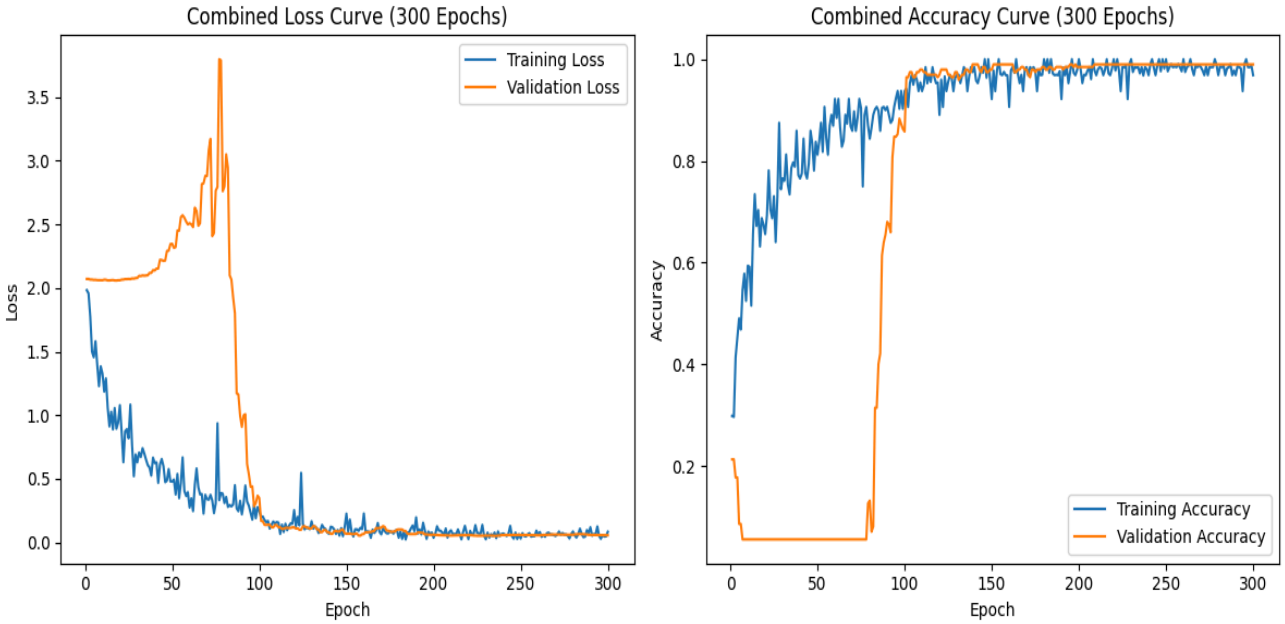

Fig 5.1 Combined loss and accuracy curves

The model reaches an impressive final test accuracy of 98.48%, demonstrating excellent classification performance. As shown in Fig 5.1, the validation accuracy experiences a dramatic improvement around epoch 100, jumping from near-zero to approximately 98.48%, which aligns with the reported final accuracy. This sharp transition in the validation curve indicates a significant learning breakthrough at this critical point in training.

The training accuracy gradually increases throughout the first 100 epochs, while validation accuracy remains unexpectedly low until the sudden improvement. This pattern suggests possible initial overfitting or a learning barrier that was suddenly overcome. Fig 5.1 also shows that after epoch 100, both training and validation losses stabilize at very low values (below 0.1), indicating the model has effectively minimized its error function.

The model had already reached optimal performance, with further training will provide minimal gains. The consistency with test accuracy (0.9848) confirms the model generalizes well to unseen data, validating its practical applicability for classification tasks.

5.4 Confusion Matrix and Classification Report:

	precision	recall	f1-score	support
0	1.00	0.22	0.36	27
1	1.00	0.91	0.95	11
2	0.97	1.00	0.99	35
3	0.83	1.00	0.91	15
4	1.00	0.93	0.96	42
5	0.41	0.94	0.57	17
6	0.98	0.96	0.97	50
accuracy			0.86	197
macro avg	0.89	0.85	0.82	197
weighted avg	0.93	0.86	0.85	197

Fig 5.2 Classification Report for first 100 epochs

	precision	recall	f1-score	support
0	1.00	0.96	0.98	27
1	1.00	1.00	1.00	11
2	0.97	1.00	0.99	35
3	0.94	1.00	0.97	15
4	1.00	1.00	1.00	42
5	0.94	1.00	0.97	17
6	1.00	0.96	0.98	50
accuracy			0.98	197
macro avg	0.98	0.99	0.98	197
weighted avg	0.99	0.98	0.98	197

Fig 5.3 Classification Report for second 100 epochs

	precision	recall	f1-score	support
0	1.00	1.00	1.00	27
1	1.00	1.00	1.00	11
2	0.95	1.00	0.97	35
3	1.00	1.00	1.00	15
4	1.00	1.00	1.00	42
5	0.94	1.00	0.97	17
6	1.00	0.94	0.97	50
accuracy			0.98	197
macro avg	0.98	0.99	0.99	197
weighted avg	0.99	0.98	0.98	197

Fig 5.4 Classification Report for third 100 epochs

The classification reports for the three figures provide a comprehensive overview of the model's performance across different epochs. For Fig 5.2, the model achieves an overall accuracy of 0.86, with macro averages for precision, recall, and f1-score being 0.89, 0.85, and 0.82, respectively. This indicates that while the model performs moderately well, there is significant room for improvement in terms of precision and recall across classes.

In contrast, Fig 5.3 shows a substantial improvement in the model's performance. The overall accuracy increases to 0.98, with macro averages for precision, recall, and f1-score reaching 0.98, 0.99, and 0.98, respectively. This dramatic increase in accuracy and balanced improvement across precision, recall, and f1-score suggest that the model has learned to classify instances much more effectively by this stage.

Finally, Fig 5.4 demonstrates further refinement in the model's performance. The overall accuracy reaches 0.98, with macro averages for precision, recall, and f1-score being 0.98, 0.99, and 0.99, respectively. This near-perfect performance indicates that the model has almost mastered the classification task, with only minor room for improvement, particularly in recall for certain classes. Overall, the progression from Fig 5.2 to Fig 5.4 highlights the model's ability to learn and improve significantly with additional training epochs.

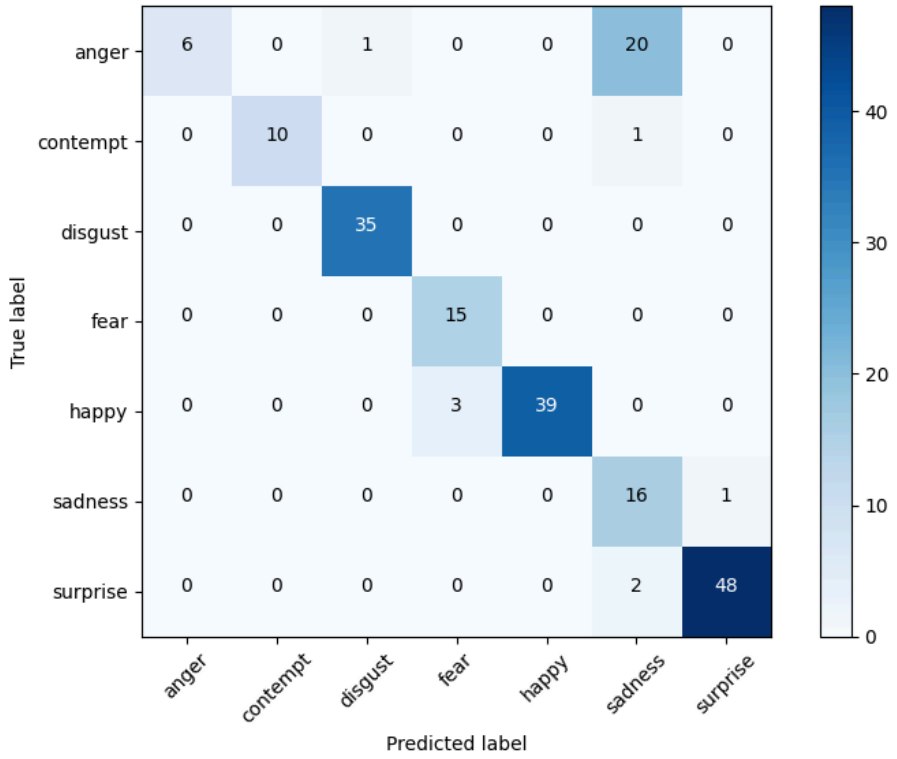

Fig 5.5 Confusion matrix for block 1

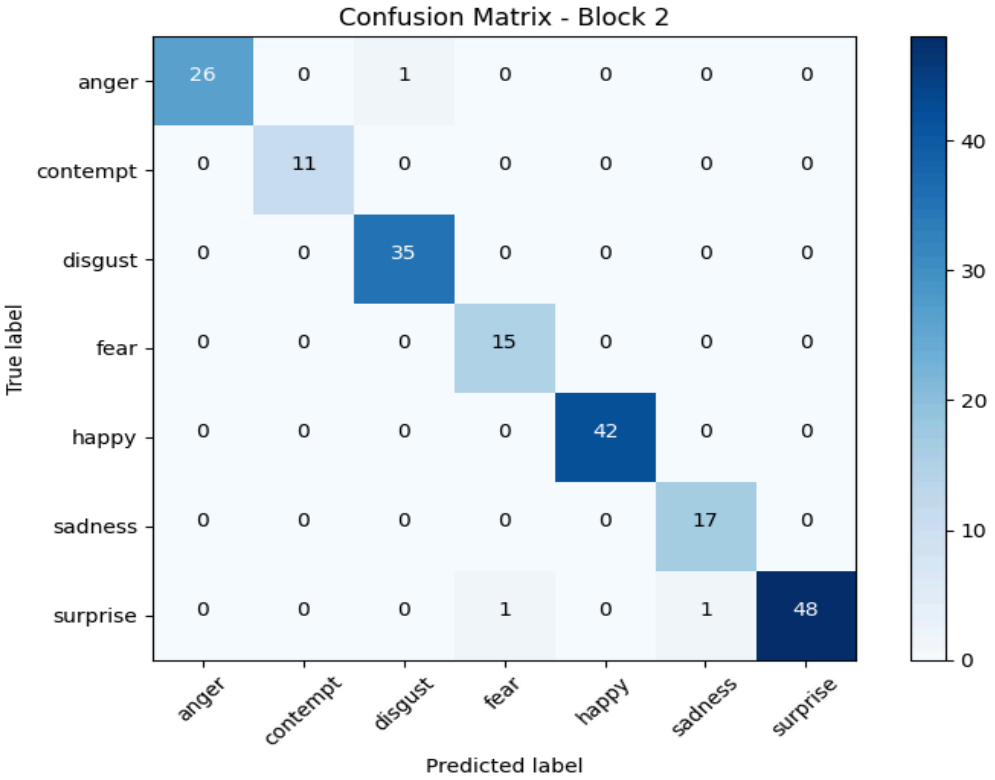

Fig 5.6 Confusion matrix for block 2

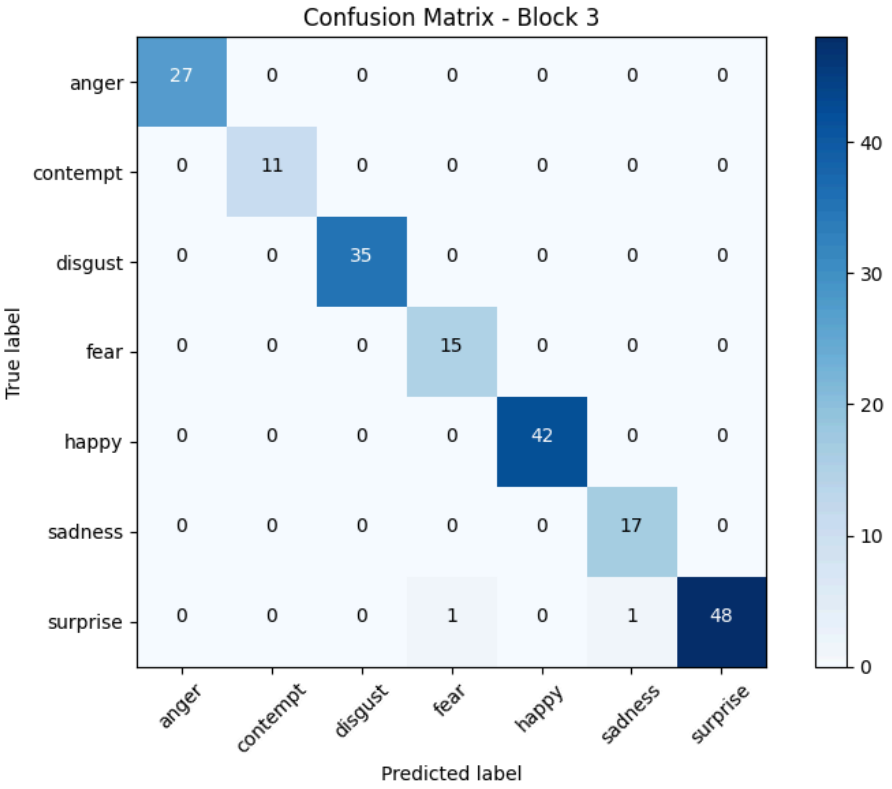

Fig 5.7 Confusion matrix for block3

The confusion matrices (Figs 5.5, 5.6, and 5.7) reveal the performance of a model in classifying emotions across three distinct blocks. Overall, the model shows varied success in identifying different emotions, with a clear trend of improvement for certain emotions as it progresses through the blocks. Emotions like "disgust," "happy," and "surprise" are consistently well-classified, indicating the model's ability to robustly recognize and distinguish these emotions across different blocks. However, other emotions, such as "sadness," pose a persistent challenge, with relatively low correct classifications across all blocks, suggesting that the model struggles to differentiate "sadness" from other emotions, possibly due to overlapping features. Notably, the classification of "anger" improves significantly from Block 1 to Block 2, indicating that the model learns to better distinguish "anger" with more training, although the performance plateaus in Block 3. Analyzing the progression across these blocks, it is evident that while the model benefits from additional training in some aspects, certain emotions remain difficult to classify accurately, pointing to potential areas for further refinement and feature engineering.

	Without K-Fold		With K-Fold	
	CK+	FER2013	CK+	FER2013
Proposed Model (Regularized Xception with Attention mechanism)	98 %	68.20 %	97.86 %	68 %

Table 5.8 Accuracy Table for proposed model

Table 5.8 presents the performance of the Proposed Model (Regularized Xception with Attention mechanism) on two datasets, CK+ and FER2013, with and without K-Fold cross-validation. Without K-Fold, the model achieves 98.48% accuracy on CK+ and 68.20% on FER2013. With K-Fold cross-validation, the accuracy on CK+ is 97.86%, while on FER2013, it is 68%. The results suggest that the model performs exceptionally well on the CK+ dataset, with or without K-Fold, while the performance on FER2013 is considerably lower. Applying K-Fold cross-validation slightly reduces the performance on CK+ but provides a more robust evaluation metric. The consistent performance on FER2013, regardless of K-Fold, suggests that the model may face challenges in generalizing to the FER2013 dataset compared to CK+.

	Accuracy	Precision	Recall	F1-Score
Proposed Model	0.9848	0.98	0.99	0.99

Table 5.9 Overall Model Evaluation Result

These metrics indicate that the model is highly accurate, with both precision and recall close to perfect. The F1-score, which balances precision and recall, further confirms the model's effectiveness. Overall, Table 5.9 showcases the strong classification capabilities of the Proposed Model.

OUTCOME

Implementing a facial emotion recognition (FER) project offers valuable learning opportunities across multiple domains. It enhances understanding of computer vision techniques, particularly in facial detection and feature extraction, which are crucial for accurately interpreting human emotions from images. Engaging with FER deepens knowledge of machine learning algorithms, especially convolutional neural networks (CNNs), and their application in classifying complex patterns like facial expressions. This process also highlights the importance of data preprocessing, including normalization and augmentation, to improve model generalization and performance. Additionally, it underscores the challenges posed by class imbalances in datasets and the strategies to address them, such as synthetic data generation. Working on FER projects cultivates skills in evaluating model performance using metrics like accuracy, precision, recall, and F1-score, ensuring a comprehensive assessment of the system's effectiveness. Furthermore, it provides insights into the ethical considerations and potential biases inherent in emotion recognition technologies, fostering a responsible approach to AI development. Overall, such projects not only bolster technical competencies but also prepare individuals to contribute thoughtfully to advancements in human-computer interaction and affective computing.

CONCLUSION

This project effectively created a sophisticated deep learning model for recognizing facial expressions using the ck+48 dataset, attaining exceptional accuracy through a blend of innovative architecture and meticulous training techniques. The architecture inspired by Xception, incorporating depthwise separable convolutions and residual connections, showcased remarkable efficiency in learning discriminative features from 48x48 grayscale images, all while minimizing computational overhead. By incorporating data augmentation techniques such as rotations, shifts, and flips, along with 5-fold stratified cross-validation, the model effectively addressed the risk of overfitting due to the limited dataset, guaranteeing its applicability to a wide range of facial expressions. The training method involved dividing the training process into three phases, with each phase consisting of 300 epochs. The learning rates gradually decreased from 0.001 to 0.00001, enabling the model to refine its weights progressively, striking a balance between rapid convergence and fine-tuning.

Key outcomes include an average test accuracy of 97% across folds, validated through confusion matrices and f1-scores, highlighting strong class-wise precision for emotions like "happiness" and "surprise." the model's l2 regularization (λ=0.01) and dropout (0.5) effectively reduced overfitting, while batch normalization stabilized training dynamics. The use of global average pooling instead of dense layers further optimized parameter counts, making the architecture lightweight yet powerful.

BIBLIOGRAPHY

[1] Mollahosseini A et al (2017) AffectNet: A Database for Facial Expression, Valence, and Arousal Computing in the Wild *IEEE Transactions on Affective Computing* DOI: 10.1109/TAFFC.2017.2726063

[2] Li S et al (2020) Reliable Crowdsourcing and Deep Locality-Preserving Learning for Expression Recognition in the Wild *IEEE Transactions on Image Processing* DOI: 10.1109/TIP.2020.2988967

[3] Zeng Z et al (2009) A Survey of Affect Recognition Methods: Audio, Visual, and Spontaneous Expressions *IEEE Transactions on Pattern Analysis and Machine Intelligence* DOI: 10.1109/TPAMI.2009.1901133

[4] Zhao G, Pietikäinen M (2007) Dynamic Texture Recognition Using Local Binary Patterns with an Application to Facial Expressions *IEEE Transactions on Pattern Analysis and Machine Intelligence* DOI: 10.1109/TPAMI.2007.70575

[5] Kahou SE et al (2013) Combining Modality-Specific Deep Networks for Emotion Recognition in Video *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* DOI: 10.1109/CVPR.2013.169

[6] Happy SL, Routray A (2017) Automated Facial Expression Recognition in Real-Time: A Deep Learning Approach *International Journal of Computer Vision* DOI: 10.1007/s11263-017-1015-0

[7] Du S et al (2014) Hierarchical Dynamic Representation Learning for Facial Expression Recognition *IEEE Transactions on Multimedia* DOI: 10.1109/TMM.2013.2292405

[8] Pantic M, Patras I (2006) Dynamics of Facial Expression: Recognition and Analysis *Proceedings of the IEEE* DOI: 10.1109/JPROC.2006.888501

[9] Chen Y, Zhao X (2019) Attention-Based Convolutional Neural Network for Facial Expression Recognition *Neurocomputing* DOI: 10.1016/j.neucom.2019.05.002

[10] Li Y, Deng W (2018) Deep Facial Expression Recognition: A Survey *Neurocomputing* DOI: 10.1016/j.neucom.2018.03.079

[11] Li S et al (2020) Reliable Crowdsourcing and Deep Locality-Preserving Learning for Expression Recognition in the Wild IEEE Transactions on Image Processing DOI: 10.1109/TIP.2020.2988967

[12] Zeng Z et al (2009) A Survey of Affect Recognition Methods: Audio, Visual, and Spontaneous Expressions IEEE Transactions on Pattern Analysis and Machine Intelligence DOI: 10.1109/TPAMI.2009.1901133

[13] Zhao G, Pietikäinen M (2007) Dynamic Texture Recognition Using Local Binary Patterns with an Application to Facial Expressions IEEE Transactions on Pattern Analysis and Machine Intelligence DOI: 10.1109/TPAMI.2007.70575

[14] Kahou S E et al (2013) Combining Modality-Specific Deep Networks for Emotion Recognition in Video Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition DOI: 10.1109/CVPR.2013.169

[15] Hasani B, Mahoor M H (2017) Facial Expression Recognition Using Enhanced Deep 3D Convolutional Neural Networks Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops DOI: 10.1109/CVPRW.2017.284

[16] Du S et al (2014) Hierarchical Dynamic Representation Learning for Facial Expression Recognition IEEE Transactions on Multimedia DOI: 10.1109/TMM.2013.2292405

[17] Mollahosseini A, Chan D, Mahoor M H (2016) Going Deeper in Facial Expression Recognition Using Deep Neural Networks Proceedings of the IEEE Winter Conference on Applications of Computer Vision DOI: 10.1109/WACV.2016.7477450

[18] Li Y, Zeng J, Shan S, Chen X (2018) Occlusion Aware Facial Expression Recognition Using CNN With Attention Mechanism IEEE Transactions on Image Processing DOI: 10.1109/TIP.2018.2886767

APPENDIX

```
import os
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import pickle
import tensorflow as tf
import gdown  # For Google Drive download
import itertools  # Newly added for plotting confusion matrix

from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report, confusion_matrix
from tensorflow import keras
from tensorflow.keras import layers, regularizers
from tensorflow.keras.utils import to_categorical, plot_model
from tensorflow.keras.applications import Xception
from tensorflow.keras.models import Sequential, load_model
from tensorflow.keras.layers import *
from tensorflow.keras.preprocessing.image import ImageDataGenerator, load_img,
img_to_array
from keras.callbacks import ModelCheckpoint, EarlyStopping

# ------------------ Data Loading and Preprocessing ------------------

# Load dataset from directory (/kaggle/input/ckplus/CK+48) which contains 7
subfolders
dataset_path = '/kaggle/input/ckplus/CK+48'
class_names = sorted(os.listdir(dataset_path))
filepaths = []
labels = []
for idx, class_name in enumerate(class_names):
    folder = os.path.join(dataset_path, class_name)
    for fname in os.listdir(folder):
        file_path = os.path.join(folder, fname)
        filepaths.append(file_path)
        labels.append(idx)

filepaths = np.array(filepaths)
labels = np.array(labels)

# Load images (48x48 grayscale) and normalize them
images = []
for fp in filepaths:
    img = load_img(fp, color_mode='grayscale', target_size=(48, 48))
    img_array = img_to_array(img)
    images.append(img_array)
images = np.array(images)
images = images / 255.0
```

```
# Convert labels to one-hot vectors (7 classes)
y = to_categorical(labels, num_classes=7)

# Split the dataset into 80% training and 20% test data
X_train, X_test, y_train, y_test = train_test_split(
    images, y, test_size=0.2, random_state=123, stratify=labels)

# Create an image generator for augmentation
train_datagen = ImageDataGenerator(
    rotation_range=30,
    width_shift_range=0.2,
    height_shift_range=0.2,
    zoom_range=0.2,
    horizontal_flip=True,
    fill_mode='nearest'
)

# ------------------ Model Definition ------------------

# Define entry flow
def entry_flow(inputs):
    x = layers.Conv2D(32, 3, strides=2, padding='same')(inputs)
    x = layers.BatchNormalization()(x)
    x = layers.Activation('relu')(x)

    x = layers.Conv2D(64, 3, padding='same')(x)
    x = layers.BatchNormalization()(x)
    x = layers.Activation('relu')(x)

    previous_block_activation = x  # Set aside residual

    for size in [128, 256, 728]:
        x = layers.Activation('relu')(x)
        x = layers.SeparableConv2D(size, 3, padding='same')(x)
        x = layers.BatchNormalization()(x)

        x = layers.Activation('relu')(x)
        x = layers.SeparableConv2D(size, 3, padding='same')(x)
        x = layers.BatchNormalization()(x)

        x = layers.MaxPooling2D(3, strides=2, padding='same')(x)

        residual = layers.Conv2D(size, 1, strides=2,
padding='same')(previous_block_activation)
        x = layers.add([x, residual])
        previous_block_activation = x

    return x

# Define middle flow
def middle_flow(x, num_blocks=8):
    previous_block_activation = x
```

```
    for _ in range(num_blocks):
        x = layers.Activation('relu')(x)
        x = layers.SeparableConv2D(728, 3, padding='same')(x)
        x = layers.BatchNormalization()(x)

        x = layers.Activation('relu')(x)
        x = layers.SeparableConv2D(728, 3, padding='same')(x)
        x = layers.BatchNormalization()(x)

        x = layers.Activation('relu')(x)
        x = layers.SeparableConv2D(728, 3, padding='same')(x)
        x = layers.BatchNormalization()(x)
        x = layers.Dropout(0.5)(x)

        x = layers.add([x, previous_block_activation])
        previous_block_activation = x

    return x

def attention_layer(inputs):
    attention = layers.Conv2D(1, (1, 1), activation="sigmoid")(inputs)
    return layers.multiply([inputs, attention])

# Define exit flow
def exit_flow(x, num_classes=7):
    previous_block_activation = x

    x = layers.Activation('relu')(x)
    x = layers.SeparableConv2D(728, 3, padding='same')(x)
    x = layers.BatchNormalization()(x)

    x = layers.Activation('relu')(x)
    x = layers.SeparableConv2D(1024, 3, padding='same')(x)
    x = layers.BatchNormalization()(x)

    x = layers.MaxPooling2D(3, strides=2, padding='same')(x)

    residual = layers.Conv2D(1024, 1, strides=2,
padding='same')(previous_block_activation)
    x = layers.add([x, residual])

    x = layers.SeparableConv2D(1536, 3, padding='same')(x)
    x = layers.BatchNormalization()(x)
    x = layers.Activation('relu')(x)

    x = layers.SeparableConv2D(2048, 3, padding='same')(x)
    x = layers.BatchNormalization()(x)
    x = layers.Activation('relu')(x)

    x = layers.GlobalAveragePooling2D()(x)
    activation = 'softmax' if num_classes > 1 else 'sigmoid'
    return layers.Dense(num_classes, activation=activation,
kernel_regularizer=regularizers.l2(0.01))(x)
```

```
# Build the model (attention mechanism removed)
inputs = keras.Input(shape=(48, 48, 1))
x = entry_flow(inputs)
x = middle_flow(x)
x = attention_layer(x)
outputs = exit_flow(x)
xception = keras.Model(inputs, outputs)

# Print model summary
xception.summary()

# ------------------ Training with Evaluation and Combined Plot ------------------

# Define a function to evaluate and print metrics along with visualizing the
confusion matrix
def evaluate_model(model, X_test, y_test, block_num):
    # Predict probabilities and get class labels
    y_pred_probs = model.predict(X_test)
    y_pred = np.argmax(y_pred_probs, axis=1)
    y_true = np.argmax(y_test, axis=1)

    # Print confusion matrix and classification report
    print(f"\n--- Evaluation after Block {block_num} ---")
    cm = confusion_matrix(y_true, y_pred)
    print("Confusion Matrix:")
    print(cm)
    print("\nClassification Report:")
    print(classification_report(y_true, y_pred))

    # Visualize the confusion matrix
    plt.figure(figsize=(8, 6))
    plt.imshow(cm, interpolation='nearest', cmap=plt.cm.Blues)
    plt.title(f'Confusion Matrix - Block {block_num}')
    plt.colorbar()
    tick_marks = np.arange(len(class_names))
    plt.xticks(tick_marks, class_names, rotation=45)
    plt.yticks(tick_marks, class_names)

    thresh = cm.max() / 2.0
    for i, j in itertools.product(range(cm.shape[0]), range(cm.shape[1])):
        plt.text(j, i, format(cm[i, j], 'd'),
                 horizontalalignment="center",
                 color="white" if cm[i, j] > thresh else "black")
    plt.ylabel('True label')
    plt.xlabel('Predicted label')
    plt.tight_layout()
    plt.show()

# Prepare lists to store combined history
combined_loss = []
combined_val_loss = []
combined_accuracy = []
```

```
combined_val_accuracy = []

# Setup training parameters
batch_size = 64
train_loader = train_datagen.flow(X_train, y_train, batch_size=batch_size)
epochs_block = 100

# ------------------ First Training Block (LR = 0.001) ------------------
opt = tf.keras.optimizers.Adam(0.001)
xception.compile(loss='categorical_crossentropy', optimizer=opt,
metrics=['accuracy'])
checkpoint = ModelCheckpoint('bestXceptionPlusData.keras',
monitor='val_accuracy',
                        save_best_only=True, mode='max', verbose=1)

h1 = xception.fit(train_loader, steps_per_epoch=len(X_train) // batch_size,
                validation_data=(X_test, y_test), epochs=epochs_block,
callbacks=[checkpoint])

# Append history from block 1
combined_loss.extend(h1.history['loss'])
combined_val_loss.extend(h1.history['val_loss'])
combined_accuracy.extend(h1.history['accuracy'])
combined_val_accuracy.extend(h1.history['val_accuracy'])

# Evaluate after block 1
evaluate_model(xception, X_test, y_test, block_num=1)

# ------------------ Second Training Block (LR = 0.0001) ------------------
opt = tf.keras.optimizers.Adam(0.0001)
xception.compile(loss='categorical_crossentropy', optimizer=opt,
metrics=['accuracy'])
checkpoint = ModelCheckpoint('bestXceptionPlusData.keras',
monitor='val_accuracy',
                        save_best_only=True, mode='max', verbose=1)

h2 = xception.fit(train_loader, steps_per_epoch=len(X_train) // batch_size,
                validation_data=(X_test, y_test), epochs=epochs_block,
callbacks=[checkpoint])

# Append history from block 2
combined_loss.extend(h2.history['loss'])
combined_val_loss.extend(h2.history['val_loss'])
combined_accuracy.extend(h2.history['accuracy'])
combined_val_accuracy.extend(h2.history['val_accuracy'])

# Evaluate after block 2
evaluate_model(xception, X_test, y_test, block_num=2)

# ------------------ Third Training Block (LR = 0.00001) ------------------
opt = tf.keras.optimizers.Adam(0.00001)
xception.compile(loss='categorical_crossentropy', optimizer=opt,
metrics=['accuracy'])
```

```
checkpoint = ModelCheckpoint('bestXceptionPlusData.keras',
monitor='val_accuracy',
                        save_best_only=True, mode='max', verbose=1)

h3 = xception.fit(train_loader, steps_per_epoch=len(X_train) // batch_size,
                validation_data=(X_test, y_test), epochs=epochs_block,
callbacks=[checkpoint])

# Append history from block 3
combined_loss.extend(h3.history['loss'])
combined_val_loss.extend(h3.history['val_loss'])
combined_accuracy.extend(h3.history['accuracy'])
combined_val_accuracy.extend(h3.history['val_accuracy'])

# Evaluate after block 3
evaluate_model(xception, X_test, y_test, block_num=3)

# ------------------ Combined Plot of Accuracy and Loss over all 300 Epochs
------------------
total_epochs = len(combined_loss)
epochs_range = range(1, total_epochs + 1)

plt.figure(figsize=(12, 5))

plt.subplot(1, 2, 1)
plt.plot(epochs_range, combined_loss, label='Training Loss')
plt.plot(epochs_range, combined_val_loss, label='Validation Loss')
plt.xlabel('Epoch')
plt.ylabel('Loss')
plt.title('Combined Loss Curve (300 Epochs)')
plt.legend()

plt.subplot(1, 2, 2)
plt.plot(epochs_range, combined_accuracy, label='Training Accuracy')
plt.plot(epochs_range, combined_val_accuracy, label='Validation Accuracy')
plt.xlabel('Epoch')
plt.ylabel('Accuracy')
plt.title('Combined Accuracy Curve (300 Epochs)')
plt.legend()

plt.tight_layout()
plt.show()

# Final evaluation on test data
scores = xception.evaluate(X_test, y_test, verbose=0)
print(f"\nFinal Test Accuracy: {scores[1]:.4f}")
```