Scaled-down EfficientNet for Facial Expression Recognition

Luís Miguel Aires Marques da Silva Fontes

Dissertação para obtenção do Grau de Mestre em
Engenharia Eletrotécnica e de Computadores
(2º ciclo de estudos)

Orientador: Prof. Doutor António Manuel Gonçalves Pinheiro

Outubro de 2023.

Declaração de Integridade

Eu Luís Fontes, que abaixo assino, estudante com número de inscrição m10358 de Engenharia Eletrotécnica e de Computadores da Faculdade de Engenharia, delcaro ter desenvolvido o presente trabalho e elaborado o presente texto em total consonância com o **Código de Integridade da Beira Interior.**

Mais concretamente afirmo não ter incorrido em qualquer das variedades de Fraude Académica, e que aqui declaro conhecer, e que em particular atendi à exigida referenciação de frases,extratos ,imagens e outras formas de trabalho intelectural, e assim assumo na íntegra as responsabilidades da autoria.

Universidade da Beira Interior, Covilhã 9/10/2023

Luís Miguel Alves Marques da Silva Fontes

Dedication

I wish to express my profound gratitude to everyone who helped me bring this project to fruition.

To my family, especially my mom. Your unconditional love and countless sacrifices have been the driving force behind my academic journey. Thank you for believing in my dreams and being by my side, even during the most challenging moments.

To my supervisors Prof. Doutor António Pinheiro and Rafael Rodrigues, for all your patience, guidance and comprehension. Your availability to discuss solutions and sharing your insights and expertise provided invaluable feedback. Your mentorship has undeniably improved the quality of my work.

To my friend João Prazeres, for his advice and technical assistance. Your companionship provided much-needed respite and moments of enjoyment.

And to my girlfriend Daniela Castellanos, for your unwavering support and affection. Your encouragement and belief in my potential have been a constant source of inspiration. You have stood by me through the long nights and demanding days.

This dissertation represents the culmination of a challenging and rewarding journey, and it would not have been possible without the help of each of you.

Resumo

O Reconhecimento de Expressões Faciais (FER) automatizado atraiu interesse significativo ao longo dos últimos anos no campo de visão computacional, com uma ampla gama de aplicações de vão desde interação humano-máquina, até segurança, educação e saúde. As técnicas de transferência de aprendizagem permitem utilizar modelos treinados para reconhecer padrões num dado domínio/tarefa e utilizar os parâmetros aprendidos para melhorar o seu desempenho numa tarefa diferente. Esta tornou-se a forma mais rápida e amplamente adotada para treinar redes neuronais convolucionais(CNNs) para a tarefa de FER. Apesar desta ser atualmente a forma mais eficiente e adotada, vale a pena considerar a possibilidade se uma rede treinada do zero para FER obteria melhores resultados.
Esta dissertação explora o desenvolvimento de uma CNN especializada para tarefas de FER sem depender de técnicas de transferência de aprendizagem. O sistema Optuna foi utilizado para procurar uma configuração ótima de hiperparâmetros para uma arquitectura EfficientNetV2S reduzida treinada do zero, notavelmente os coeficientes de profundidade e largura que afectam o número de camadas e de filtros da CNN.
Uma experiência de referência foi inicialmente efectuada na base de dados FER2013, utilizando a arquitetura original do modelo EfficientNetV2S com o tamanho de entrada ajustado para corresponder ao tamanho das imagens da base de dados. Os resultados obtidos nesta experiência revelaram um nível de desempenho comparáveis à exatidão humana e próxima do desempenho estado-da-arte. A experiência reduzida subsequente melhorou consideravelmente a eficácia da CNN reduzindo o número de parâmetros e tempo de treino enquanto mantém um nível de desempenho semelhante à experiência de referência. Em ambas as experiências a exatidão global foi cerca de 66% para os dados de teste FER2013. A experiência reduzida também foi efectuada nos dados de test da AffectNet, alcançando a exatidão global de 61,3%. É de notar que estes resultados referem-se a um problema de 7 classes em bases dados desequilibradas. Para classes individuais, a experiência reduzida alcançou uma exatidão de 85%(FER2013) e 86%(AffectNet) para amostras de teste rotuladas como *Feliz* e 81% para amostras de teste rotuladas *Surpresa* no AffectNet.

Palavras-chave

Reconhecimento de Expressões Faciais, *Deep Learning*, EfficientNet, Otimização de Hiperparâmetros

Abstract

Automated Facial Expression Recognition (FER) has attracted significant interest over the last few years in the field of computer vision, with a wide range of applications ranging from human-machine interaction to security, education, and healthcare. Transfer learning techniques allow to use models trained for pattern recognition in a given domain/-task and use their learned features to enhance their performance in a different task. This has become the fastest and most adopted way to train Convolutional Neural Networks (CNNs) for the task of FER. Although this is currently the most adopted and efficient way, it is worth considering whether a model trained from scratch for FER would yield better results.

This dissertation explores the development of a specialized deep learning CNN for FER tasks without relying on transfer learning techniques. The Optuna framework was used to search for an optimal configuration of hyperparameters for a scaled-down EfficientNetV2S architecture trained from scratch, notably the depth and width coefficients affecting the number of layers and filters of the CNN.

A baseline experiment was initially performed on the FER2013 dataset using the original EfficientNetV2S model architecture, with the input size adjusted to match the image size of the dataset. The results obtained in this experiment revealed performance levels comparable to human accuracy and near state-of-the-art performance. The subsequent scaled-down experiment considerably increased the efficiency of the CNN by reducing the number of parameters and training time while maintaining performance levels close to those of the baseline experiment. In both experiments, the obtained global accuracy was around 66% for the FER2013 test data. The scaled-down experiment was also performed with the AffectNet test data, achieving a global accuracy of 61.3%. It should be noted that these results refer to a 7-class problem on highly imbalanced datasets. For individual classes, the scaled-down experiment reached an accuracy of 85% (FER2013) and 86% (AffectNet) for test samples labeled as *Happy*, and 81% for *Surprise* test samples of AffectNet.

Keywords

Facial Expression Recognition, Deep Learning, EfficientNet, Hyperparameter Optimization

Contents

List of Figures

List of Tables

Acronyms and Abbreviations

AI	Artificial Intelligence
ANN	Artificial Neural Network
AU	Action Unit
CNN	Convolutional Neural Network
DL	Deep Learning
ELU	Exponential Linear Unit
FACS	Facial Action Coding System
FR	Face recognition
FER	Facial Expression Recognition
GAN	Generative Adversarial Network
HoG	Histogram of Gradients
ILSVRC	ImageNet Large Scale Visual Recognition Challenge
ML	Machine Learning
NAS	Neural Architecture Search
SGD	Stochastic Gradient Descent
SE	Squeeze and Excitation
PAD	Pleasure Arousal Dominance
ReLU	Rectified Linear Unit
RGB	Red Green Blue
HoG	Histogram of Gradients
UBI	Universidade da Beira Interior

Chapter 1

Introduction

1.1 Motivation

Automated Facial Expression Recognition (FER) has applications across multiple domains, including human-computer interaction, healthcare, and security.
The ability to decipher and interpret human emotions from facial expressions is a fundamental and powerful facet of human communication.
The motivation behind this research stems from the recognition that emotions are a fundamental aspect of human communication and interaction.
Being able to equip machines with the capacity to understand and respond to human emotions opens doors to more empathetic and responsive Artificial Intelligence(AI) systems.

1.2 Objective

This research is centered on the development and exploration of a scaled-down variant of the EfficientNet architecture tailored specifically for FER. EfficientNet, renowned for its exceptional efficiency and performance in computer vision tasks, forms the foundation for my efforts. By scaling down this architecture, we endeavor to strike a balance between computational efficiency and expressive power, tailored to the unique requirements of FER.

1.3 Dissertation Outline

This dissertation is structured as follows:

- Chapter 1: This chapter outlines the motivation, objectives, and primary challenges addressed within this dissertation.
- Chapter 2, ”Core Concepts,” lays the foundation for understanding the key principles that underpin the research in this dissertation. This chapter describes several concepts in the fields of deep learning and facial expression recognition, providing readers with the necessary background knowledge.
- Chapter 3, ”State-of-the Art of Facial Expression Recognition,” delves into the current landscape of research and advancements in the field of facial expression recognition (FER). This chapter provides valuable insights into key architectures, evaluation metrics, and the latest trends, setting the stage for a comprehensive understanding of the state-of-the-art methods employed in FER.

- Chapter 4, ”Methodology,” provides a detailed insight into the research methodology and experimental setup employed in the study. This chapter outlines the research hypothesis, datasets used, architectural considerations, and the experimental procedures followed.
- Chapter 5, ”Experimental Results,” presents the outcomes and findings of the conducted experiments. This chapter explores the comparisons between unweighted and weighted loss, the results of the baseline experiment on FER2013, the complexity reduction experiments on both FER2013 and AffectNet, and a comparison to state-of-the-art methods.
- Chapter 6, ”Conclusion,” provides a summary of the accomplishments and contributions made in the study, then provides insights into potential directions for future research.

Chapter 2

Core Concepts

This chapter provides an overview of some of the main ideas behind deep learning and facial emotion recognition. Readers will have a good understanding of the ideas behind Facial Expression Recognition(FER) and Deep Learning(DL) by the end of this chapter.

2.1 Emotion Classification

Facial expressions have a high degree of variability depending on cultural differences, individual emotional states, context, or environmental factors.
However, they are recognizable by people of different cultures or languages, supporting the hypothesis that they are universal for all individuals regardless of ethnic or cultural differences.
Also, according to discrete basic emotion theory, humans possess an innately small number of basic emotions that are cross-culturally recognizable.

2.1.1 Ekman's universal basic emotions

Dr. Paul Ekman [1] identified the six basic emotions as anger, surprise, disgust, enjoyment, fear, and sadness, with evidence of a seventh emotion, contempt. Additionally, each emotion acts as a discrete category rather than an individual emotional state. Individuals can exhibit particular characteristics attached to each of these emotions, allowing them to be expressed in varying degrees.

2.1.2 Dimensional models

Contrary to the implications of the discrete basic emotions theory, emotions seem to exist along a continuum, devoid of clear-cut boundaries that definitively distinguish one emotion from another.
Dimensional models of emotion contend that affective states result from a cognitive interpretation of fundamental neural sensations produced by various neurophysiological systems, in contrast to theories that propose discrete and independent neural systems for each emotion [2].

2.1.2.1 Circumplex model

The circumplex model of affect, also known as the circumplex model of emotion, is a psychological framework that organizes human emotions into a two-dimensional circular

space based on their underlying dimensions. This model provides a way to visualize and understand the structure of emotions by mapping them onto a circular graph.
The key idea is that emotions can be described and differentiated based on two primary dimensions: valence and arousal.
Valence refers to the pleasantness or unpleasantness of an emotion. This dimension ranges from positive to negative. Emotions on the positive end of the valence spectrum are typically associated with positive affect, such as happiness, joy, and love. Emotions on the negative end are associated with negative affect, such as sadness, anger, and fear. In the circumplex model, valence is represented on the horizontal axis, with positive emotions to the right and negative emotions to the left.
Arousal represents the level of activation or intensity of an emotion. Emotions that are highly arousing are intense and energetic, while those with low arousal are calm and passive. Emotions like excitement, anger, and fear are considered high-arousal emotions, whereas calmness, relaxation, and contentment are low-arousal emotions. In the circumplex model, arousal is represented on the vertical axis, with high arousal at the top and low arousal at the bottom.

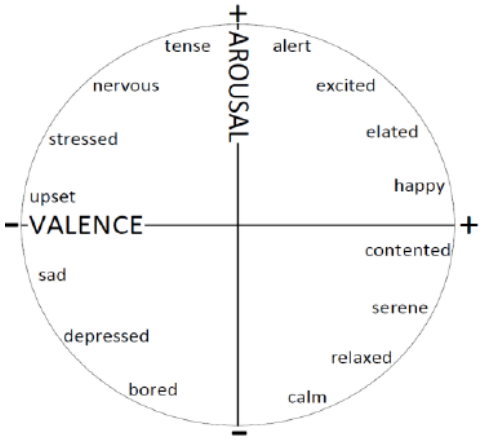

Figure 2.1: Circumplex model of affect, the x axis corresponds to valence and y axis corresponds to arousal.

2.1.2.2 PAD model

The Pleasure Arousal Dominance (PAD) model [3] constitutes a foundational psychological framework designed to elucidate and quantify emotional states. This model employs a triad of numerical dimensions, namely pleasure, arousal, and dominance, to comprehensively delineate the spectrum of human emotions.
The Pleasure-Displeasure Scale serves as a metric for gauging the degree of pleasantness associated with a given emotional state. To illustrate, emotions such as anger and fear, both categorized as unpleasant, exhibit elevated scores on the displeasure scale, whereas joy embodies a salient example of a pleasant emotional state.
Conversely, the Arousal-Nonarousal scale functions as a tool to measure the level of alertness or lethargy engendered by a particular emotional state. Not to be confused with the

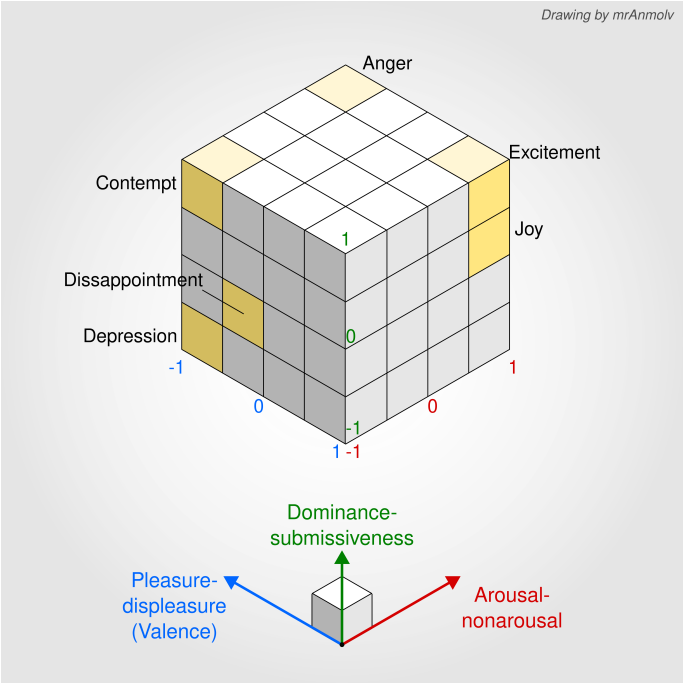

Figure 2.2: PAD emotional state model.

intensity of the emotion itself, it pertains to the state of arousal or energy. As an illustration, emotions like anger and rage, both characterized as unpleasant, manifest distinctive arousal levels, with rage featuring a heightened intensity or a state of elevated arousal. In contrast, boredom, also an aversive emotional state, registers a notably low arousal value.

The Dominance-Submissiveness scale within the PAD model delineates the assertive and submissive attributes inherent to an emotion. For instance, despite the fact that both fear and anger fall under the category of unpleasant emotions, anger is typified as a dominant emotion, whereas fear is ascribed a submissive quality, signifying their respective positions along the dominance-submissiveness continuum.

2.2 Facial Expression Recognition

The Facial Action Coding System (FACS) is a widely adopted methodology for providing comprehensive and objective descriptions of facial expressions. It achieves this by assigning specific codes to the movements of facial muscle groups known as action units (AUs).

By deconstructing facial expressions related to emotions into their individual AUs and evaluating the intensity of these AUs, domain experts can accurately assess and discern the underlying emotion being expressed. This meticulous approach to annotation is commonly applied to datasets, facilitating the precise categorization of facial expressions.

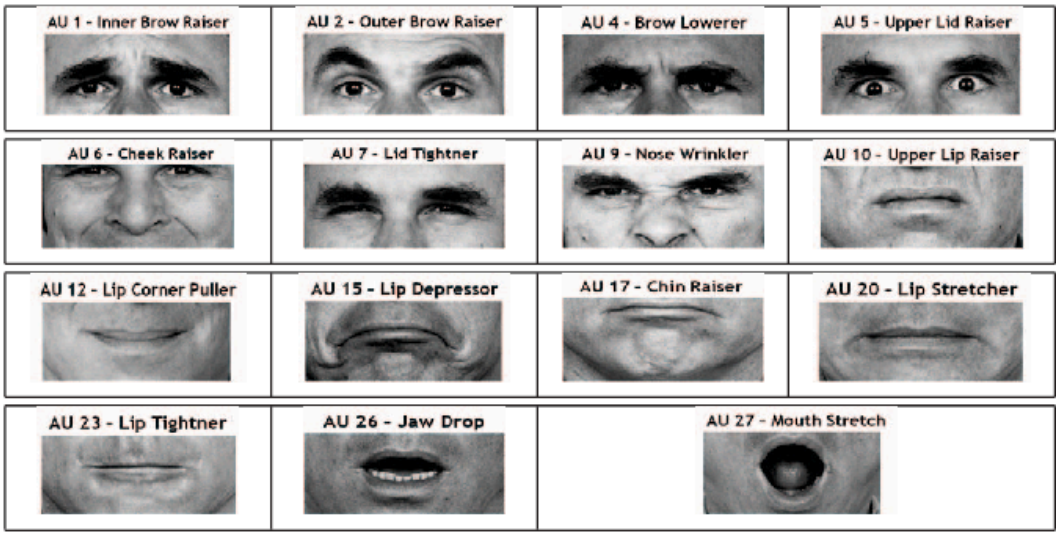

Figure 2.3: Some AU examples.

Table 2.1: Basic emotions and respective AUs with FACS codes in brackets.

Emotion	**Associated AUs**
Happiness	Cheek Raiser(6), Lip Corner Puller(12)
Sadness	Inner Brow Raiser(1), Brow Lowerer(4), Lip Corner Depressor(15)
Surprise	Inner Brow Raiser(1), Outer Brow Raiser(2), Upper Lid Raiser(5), Jaw Drop(26)
Fear	Inner Brow Raiser(1), Outer Brow Raiser(2), Brow Lowerer(4), Upper Lid Raiser(5), Lid Tightener(7), Lip Stretcher(20), Jaw Drop(26)
Anger	Brow Lowerer(4), Upper Lid Raiser(5), Lid Tightener(7), Lip Tightener(23)
Disgust	Nose Wrinkler(9), Lip Corner Depressor(15), Lower Lip Depressor(16)
Contempt	Lip Corner Puller(12), Dimpler(14)

2.3 Deep Learning Concepts

In order to establish a clear understanding of deep learning, it is crucial to provide concise definitions that distinguish it from artificial intelligence (AI) and machine learning (ML). Artificial intelligence is a field of computer science and engineering dedicated to the development of machines or software with the capacity to emulate human intelligence and make informed decisions based on information perceived from their environment.
Machine learning is a subfield of artificial intelligence that employs algorithms and techniques such as artificial neural networks, regression models, decision trees, support vector machines, k-nearest neighbor models, etc. to recognize patterns from data and make autonomous decisions without explicit programming. In traditional machine learning, a defining step is feature engineering, through which relevant features are extracted from raw data either through manual selection or automated processes. These features serve as inputs to the machine learning model during training and prediction.
Deep learning is a subfield of machine learning focusing on artificial neural networks that consist of multiple layers of neurons, hence the term “deep” in deep learning. This allows the network to learn and extract increasingly complex features and patterns from the input

data and create hierarchical representations, from lower-level features like edge detection to more sophisticated and abstract higher-level features like image recognition. Another difference from traditional machine learning is that data can be directly input into the artificial neural network, which is responsible for its own feature extraction.

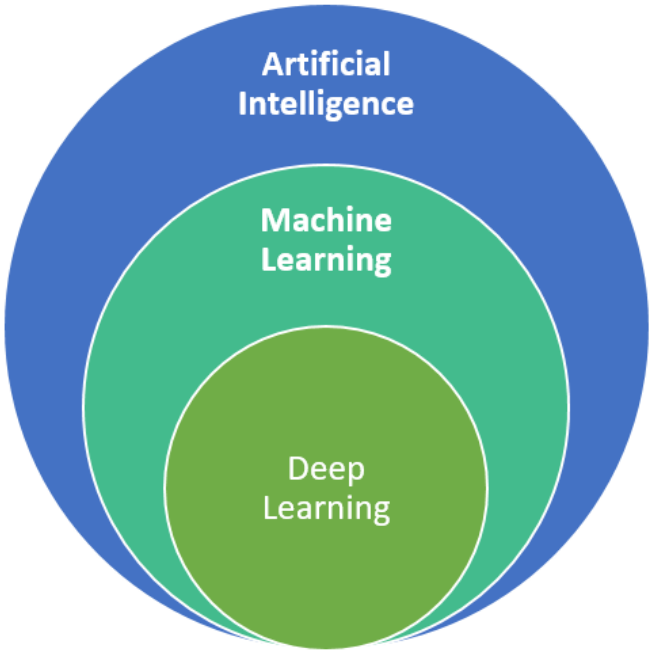

Figure 2.4: Venn Diagram of AI, ML and DL.

2.3.1 Machine Learning Paradigms

2.3.1.1 Supervised Learning

In supervised learning, the model is trained on a labeled dataset, meaning each data input has a corresponding target label. During training, the algorithm iteratively adjusts its parameters to minimize the difference between its predictions and the true labels, thus learning a mapping between input features and the target labels. The model then makes predictions on new, unseen data based on its training. Common tasks in supervised learning include classification, where the algorithm predicts a discrete label for each input, and regression, where it predicts a continuous value. Some examples include image recognition, speech recognition, natural language processing, etc.

2.3.1.2 Unsupervised Learning

In unsupervised learning, the model is not given any target labels during training. Its goal is to identify patterns, structures, or relationships within the data without any specific guidance. Some tasks include clustering, where the model groups similar data points into clusters based on their features.

2.3.1.3 Reinforcement Learning

In reinforcement learning, an intelligent agent learns how to make decisions based on its interactions with an environment, which provides feedback to the agent in the form of rewards and penalties. The agent must then develop a strategy to maximize the cumulative rewards over time. Reinforcement learning is suited for situations where the consequences of actions are delayed and feedback is infrequent or limited. These include playing games, robotics, autonomous systems, finance, etc.

2.4 Artificial Neural Network

Artificial Neural Networks draw inspiration from the functioning of biological neurons.

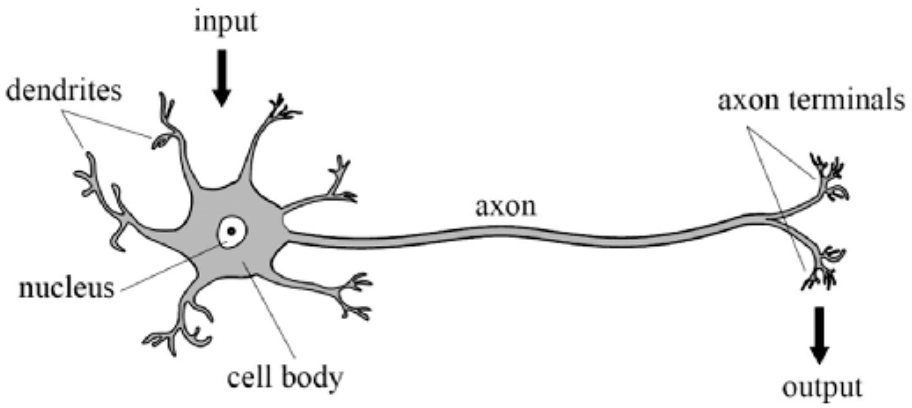

Figure 2.5: Model of a biological neuron.

In a biological brain, neurons receive signals from other neurons and possess an electrochemical threshold that determines whether the accumulated information is sufficient to trigger the neuron's activation. Furthermore, more or stronger input signals usually cause ion channels to conduct electricity better or make signal propagation easier, which can either make each connection stronger or weaker.
In artificial neural networks, the mathematical neuron attempts to emulate this behavior by multiplying each input signal with a weight term and adjusting its activation threshold using a bias term.
The multilayer perceptron is the simplest example of an artificial neural network consisting of an input layer, a number of hidden layers, and an output layer.

2.4.1 Activation functions

The most important aspects of activation functions are non-linearity, continuity, and differentiability. Incorporating non-linearity is essential, considering the combination of linear functions yields a linear function. Any problem that can be solved by a polynomial function would be trivially solved by analytical methods. The use of non-linear activation

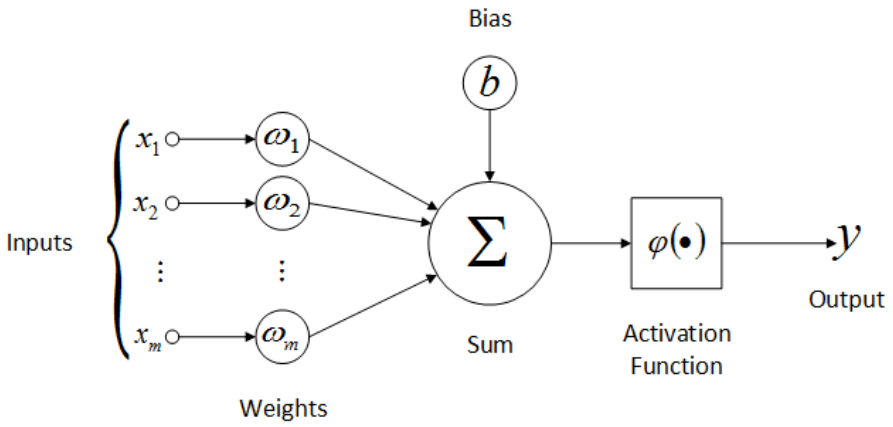

Figure 2.6: Model of a mathematical neuron.

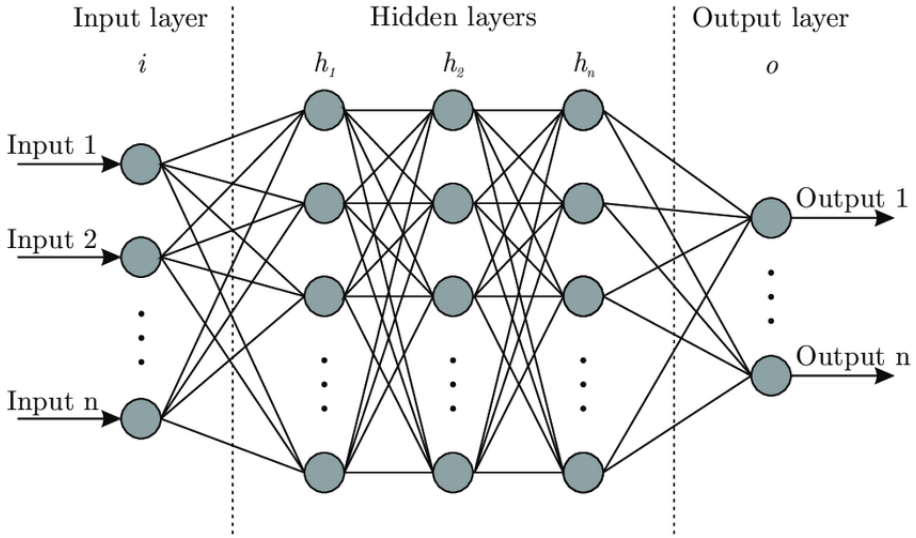

Figure 2.7: Multilayer Perceptron.

functions enables models to approximate complex relationships in data according to the universal approximation theorem, which states that with just a single hidden layer and non-linear activations, given sufficient neurons, it is possible to approximate any continuous function to arbitrary precision. Furthermore, with progressively deeper layers, the model gains the capacity to learn hierarchical representations of features in the data and build high-level abstractions upon the lower-level features extracted in the previous layers.
Continuity ensures smooth output transitions, enabling gradient-based optimization. Differentiability is mandatory for the backpropagation algorithm used for adjusting parameters during training.

2.4.1.1 Sigmoid

The sigmoid function is one of the most used activation functions in machine learning and deep learning. It is known for taking any real value input and outputting value between 0

and 1.

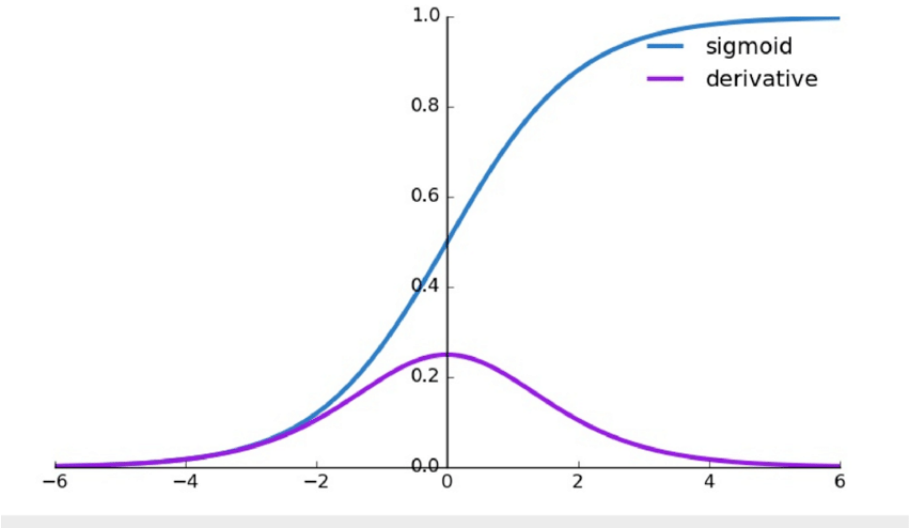

Figure 2.8: Sigmoid and its derivative.

2.4.1.2 ReLU

The most common activation function used in deep neural networks is the ReLU (rectified linear unit), defined as:

$$\text{ReLU}(x) = \begin{cases} x & \text{if } x > 0 \\ 0 & \text{if } x \leq 0 \end{cases} \tag{1}$$

Its output is defined as 0 for negative inputs or output equal to input for positive numbers.

The distinguishing feature of this function is that its derivative is either 0 for negative inputs or 1 for positive inputs, simplifying computations.

The main drawback of this approach is its inherent vulnerability. Specifically, when undergoing training, the utilization of gradient-based updates may result in certain inputs traversing into the negative activation zone, causing the neuron to produce an output of zero. When neurons in the network provide 0 values to the predictions, their weights do not undergo significant updates throughout the backpropagation process. This is because the gradients associated with negative inputs are zero. This leads to the neurons being inactive during the training process, giving rise to a phenomena commonly referred to as the ”dying ReLU” problem. In order to address this issue, various alternative activation functions such as Swish, Leaky ReLU, and Exponential Linear Unit (ELU) may be employed. The utilization of these activation functions incorporates a marginal non-zero gradient for negative inputs, so mitigating the occurrence of ”dying” behavior and enabling neurons to acquire knowledge even in instances where their activations are negative.

2.4.1.3 Swish

Swish [4] is an alternative activation function to ReLU designed to mitigate the "dying ReLU" problem while offering a continuous and smooth derivative across its entire domain. However, this advantage comes at the cost of increased computational complexity due to the additional multiplication operation. Swish is mathematically defined as:

$$\text{Swish}(x) = x \cdot \sigma(\beta x) = \frac{x}{1 + e^{-\beta x}} \tag{2}$$

Swish introduces a hyperparameter denoted as β, which is often maintained at a value of 1, resulting in what is referred to as Swish-1, equivalent to the SiLU [5] function.

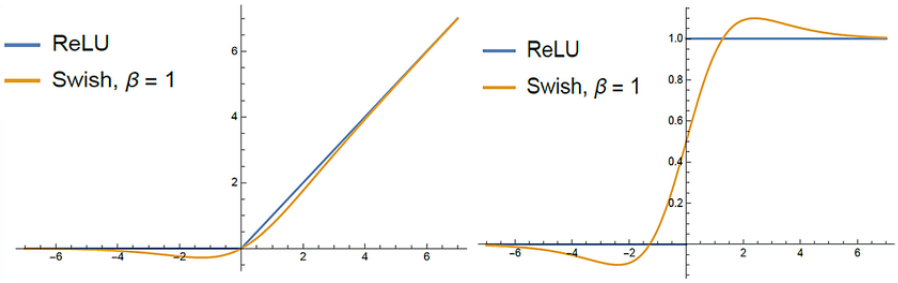

Figure 2.9: ReLU, Swish and respective derivatives.

2.4.1.4 Softmax

The softmax function is used in the final classification layer for its ability to transform a vector of logits z into a multi-class probability distribution p, defined as:

$$\hat{y}_i = \frac{e^{z_i}}{\sum_i e^{z_i}} \tag{3}$$

Where $\hat{y}_i$ is the predicted probability that a given input data point belongs to class i. And z_i is the logit associated with class i.
Logits are the raw, unnormalized scores or activations generated by the model before applying the softmax function.
The softmax function, or normalized exponential function, is applied to the logits (raw scores) of each class in the final classification layer of a neural network, mapping them into a probability distribution where each value represents the probability of the input belonging to a specific class while ensuring the sum of all probabilities across classes is equal to 1. Additionally, its differentiability is also a requisite for the backpropagation algorithm. Other desirable properties include its synergy with the cross-entropy loss function, a topic to be elaborated upon in the subsequent section.

2.4.2 Loss Function

The loss function (often denoted as L) serves to evaluate the performance of an ANN by quantifying the error between the model's predictions and the actual target values.

For this reason, in order to attain the optimal level of performance for the ANN, the optimizer seeks to find the minimum of this loss function.
The multiclass categorical crossentropy loss function is defined as:

$$L = -\sum_{i=1}^{C} [y_i \cdot \log(p_i)] \tag{4}$$

Where C is the number of classes, y_i is the true label encoded in a one-hot vector, and p_i is the predicted probability for each class.
The derivative of the softmax function nicely cancels out most terms in the gradient expression, leaving a relatively simple and efficient expression.

$$\frac{\partial L}{\partial z_i} = \hat{y}_i - y_i \tag{5}$$

The computation time for backpropagation is much lower with this simplified gradient expression because it only needs to do simple additions and subtractions between the predicted probabilities p and the true target probabilities y.

2.4.3 Optimizer

The goal when training a neural network is to find the minimum of the loss function, also known as the error function. Searching for the optimal values of the weights and biases that minimize the objective function is a complex optimization task.
A general method for minimizing differentiable functions is gradient descent.
This algorithm calculates the gradient of the loss function used in artificial neural networks starting from the last layer and iteratively progresses backward through the preceding layers by virtue of the backpropagation algorithm.
The backpropagation algorithm consists of calculating the gradients of the loss function with respect to each of the network's parameters using the chain rule from calculus.

$$\frac{\partial L}{\partial W^{(l)}} = \frac{\partial L}{\partial \text{output}^{(l)}} \cdot \frac{\partial \text{output}^{(l)}}{\partial \text{input}^{(l)}} \cdot \frac{\partial \text{input}^{(l)}}{\partial W^{(l)}} \tag{6}$$

Afterwards, it adjusts these parameters in the direction opposite to the gradient, scaled by a learning rate coefficient. This iterative process is repeated every epoch.

$$\theta_{\text{new}} = \theta_{\text{old}} - \text{learning\_rate} \cdot \frac{\partial L}{\partial \theta} \tag{7}$$

The learning rate plays a crucial role in controlling the size of parameter updates. By adjusting parameters in this manner, the algorithm gradually optimizes the model's performance by guiding the parameters towards values that minimize the loss function.
Loss Function (Cost Function): The goal of gradient descent is to minimize this loss function:

$$f(\theta) = L(y, \hat{y}(\theta)) \tag{8}$$

Here, y represents the actual target values, $\hat{y}(\theta)$ is the model's prediction based on the current parameters θ, and $f(\theta)$ is the loss function.

In batch gradient descent, the gradient is computed using the entire training dataset:

$$\nabla f(\theta) = (1/n) * \sum_{i=1}^{n} [\nabla L(y_i, \hat{y}(\theta))] \tag{9}$$

for all data points (i = 1 to n) This formula calculates the average gradient over all training examples, where n is the number of data points.

2.4.3.1 Stochastic Gradient Descent

In stochastic gradient descent, the gradient is computed for a single randomly selected data point at each iteration:

$$\nabla f(\theta) = \nabla L(y_i, \hat{y}(\theta)) \tag{10}$$

for a randomly chosen data point i.

This introduces randomness and can help escape local minima.

Mini-batch stochastic gradient descent computes the gradient using a small random subset (mini-batch) of the training data at each iteration:

$$\nabla f(\theta) = (1/b) * \sum_{i=1}^{b} [\nabla L(y_i, \hat{y}(\theta))] \tag{11}$$

for a mini-batch of size b.

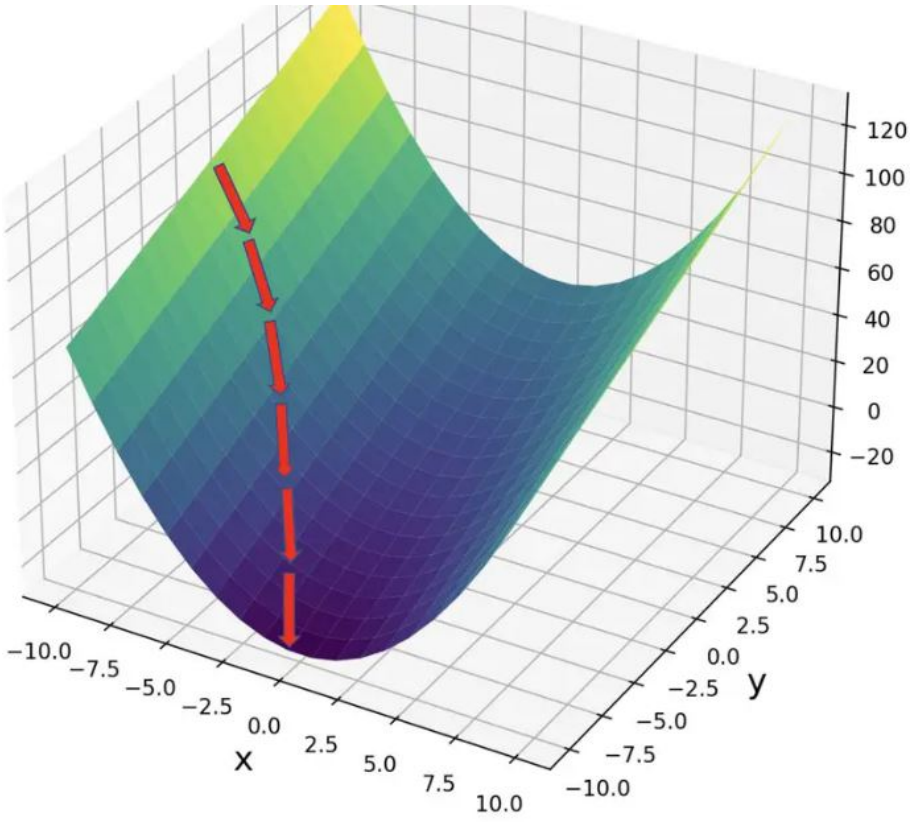

Figure 2.10: Representation of Gradient Descent.

2.4.3.2 Adam

Adam [6] (short for Adaptive Moment Estimation) is a popular optimization algorithm used to update the parameters (weights and biases) of a neural network during the training process.

It's an extension of the stochastic gradient descent (SGD) optimization algorithm that combines the advantages of both gradient-based optimization and momentum-based optimization techniques. Adam is designed to adaptively adjust the learning rates for each parameter, making it well-suited for a wide range of machine learning tasks.

The hyperparameters necessitating specification within the Adam optimization algorithm encompass the learning rate, conventionally represented as α, β1 denoting the first moment exponential decay, β2 denoting the second moment exponential decay, and ε signifying a minute value employed to avert division by zero.

Adam starts by setting the model's parameters (weights and biases) and introducing two moving average variables,'m' and 'v,' both initially assigned zero vectors. These moving averages serve the purpose of storing estimates related to the first and second-order moments of the gradients, which correspond to the mean and uncentered variance.

Then the gradients of the loss function are calculated with respect to the model's parameters using backpropagation.

The moving averages 'm' and 'v' are updated through the utilization of exponential moving

averages derived from the current gradient and its squared value. Notably, this updating mechanism bears resemblance to the procedures employed within the momentum optimizer, which employs an exponentially weighted moving average to adjust the optimization process based on past gradients and their momentum.
Then bias correction steps are applied to m and v to counter a possible bias towards zero at the beginning resulting from the zero initialization of the moving averages.
Finally, the model's parameters are updated using the computed gradients in conjunction with the moving averages. The update mechanism bears resemblance to the fundamental gradient descent algorithm; however, it is distinguished by the adaptive scaling of the learning rate coefficient based on the first and second moment estimates, rendering it an instance of adaptive optimization techniques.
As Adam employs adaptive learning rates tailored to individual parameters, it enhances the optimization process's ability to effectively traverse complex optimization terrains. Consequently, this adaptation contributes to accelerated convergence toward optimal solutions.
Adam often needs less manual calibration of learning rates and other hyperparameters than other optimization algorithms. This is another benefit of using adaptive learning rates.

2.5 Bias-variance decomposition

Irreducible error is the baseline error you don't expect your model to do better. It can be estimated using strong baselines, like human performance.
Avoidable bias, a measure of underfitting, is the difference between our train error and irreducible error.
Variance, a measure of overfitting, is the difference between validation error and training error.
Validation overfitting is the difference between test error and validation error.
These concepts can be generally understood as:

$$\text{Test error} = \text{Irreducible error} + \text{Bias} + \text{Variance} + \text{Validation overfitting}$$

2.5.1 Underfitting

Underfitting is characterized by high bias and low variance. This occurs when the model is not capturing the complex relationships or underlying patterns in the data, which likely indicates the learning model is too simplistic or lacks representational capacity. The model may have an insufficient number of parameters or be too rigid. This results in poor performance both during training and evaluation. Possible solutions include increasing model complexity, increasing model capacity by adding layers or increasing the number of neurons per layer, or it could be a result of excessive regularization.

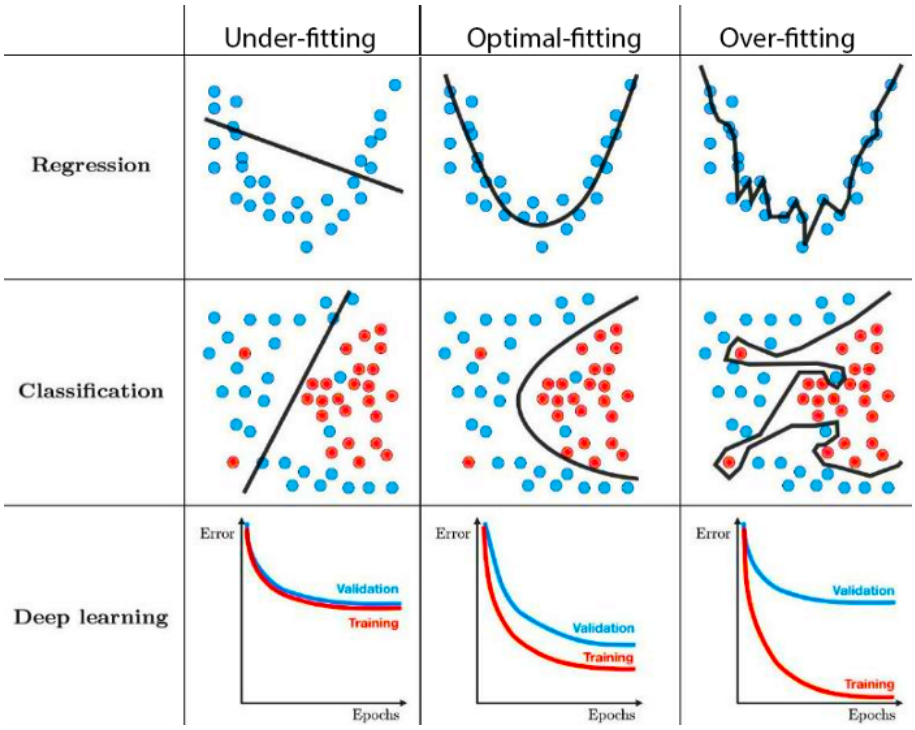

Figure 2.11: A visualization of underfitting, overfitting in Regression, Classification and Deep learning tasks.

2.5.2 Overfitting

In contrast, overfitting is characterized by low bias and high variance as a consequence of the model memorizing the noise and outliers in training data instead of learning good generalization patterns that would enable it to perform accurate predictions on unseen data. This problem is evident when the model has exceptional performance during training but performs poorly during evaluation on new data. Some solutions are reducing model complexity, increasing the amount of data, and applying regularization techniques.

2.5.3 Optimal fit

A desired goal in machine learning is to achieve a trade-off between bias and variance, in other words, striking a balance between cautious generalization and overconfident predictions. A good and robust generalization should have the right level of confidence in its predictions while being adaptable to new information.

2.6 Improvement techniques

2.6.1 Data augmentation

To enhance the generalization and robustness of the machine learning model, it should be exposed to a wider range of scenarios. Nevertheless, the restricted size of datasets often presents an obstacle to performing this approach. By employing data augmentation

techniques, it is possible to generate new variations by applying different transformations to the original samples. Training on augmented data allows the model to become invariant to certain transformations, thus reducing overfitting and improving performance on real-world data. Some examples of data augmentation techniques employed in computer vision are rotation, flipping, cropping, resizing, brightness and contrast adjustments, and color transformations.

2.6.2 Dropout

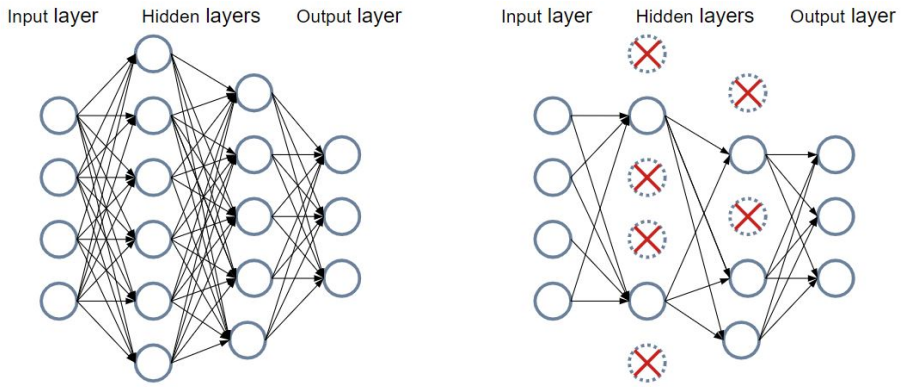

Figure 2.12: Before applying dropout(left) After applying dropout(right).

Dropout [7] is a widely adopted regularization technique for deep neural networks. Dropout serves as a countermeasure against overfitting, employing a straightforward concept. During each training iteration, a neuron is temporarily rendered inactive or ”dropped,” with a probability denoted as p. Consequently, all inputs and outputs linked to this neuron become inactive for the current iteration. The dropped neurons undergo resampling with a probability of p at each training step, allowing a neuron dropped in one step to be active in the next. The dropout rate, a crucial hyperparameter, is typically set between 0 and 0.5, equivalent to approximately 50The effectiveness of dropout is rather surprising, as it deliberately disables neurons yet leads to improved network performance. This paradox is explained by preventing the network’s reliance on a limited set of neurons and compelling each neuron to function autonomously and independently.

2.6.3 Batch Normalization

Batch normalization is a technique used in training deep neural networks that helps improve training stability and convergence speed by normalizing the activations of intermediate layers within a neural network during training. The underlying principles for batch normalization’s success are not entirely understood.
In the original work by Ioffe and Szegedy [8] it is proposed that by maintaining consistent input distributions, batch normalization helps in stabilizing the training process. It prevents vanishing and exploding gradients by normalizing the gradients as well. This, in

turn, allows for the use of higher learning rates, faster convergence, and more stable optimization. When the parameters of earlier layers change, they affect the input distribution to subsequent layers, leading to a shift in the statistical properties of the activations. This phenomenon is called internal covariate shift, and it can slow down the training process and make it more difficult for the network to converge on a good solution.
Batch normalization addresses this problem by normalizing the activations of each layer with respect to the mean and variance of the activations within a mini-batch of training examples, aiming to keep them close to a standard distribution.
Santurkar *et al.* [9] argues that batch normalization's key contribution lies in smoothing the optimization landscape by making the loss function more well-behaved and reducing the presence of sharp and steep gradients. With fewer abrupt changes in gradients, the optimization process becomes more predictable and stable, minimizing the likelihood of encountering regions with challenging geometry that might slow down progress. This results in more efficient navigation toward optimal solutions and accelerates convergence.

2.7 Convolutional neural network

Convolutional neural networks (CNNs) [10] have become the preferred choice over fully connected networks in the realm of computer vision. CNNs are meticulously crafted to apprehend the intricate spatial intricacies present in data, especially pertinent to images, where the proximity of pixels often implies a meaningful relationship. To achieve this, CNNs employ convolutional layers that apply filters to localized sections of the input data, enabling them to effectively recognize and represent spatial associations—a distinctive capability not inherent in fully connected networks.
One of the distinguishing features of CNNs lies in their parameter sharing mechanism. Within their convolutional layers, CNNs employ the same set of weights, referred to as kernels or filters, to scan the entirety of the input data. This ingenious design reduces the number of trainable parameters compared to fully connected networks, thereby significantly enhancing memory and computational efficiency.
CNNs demonstrate a notable ability to capture characteristics that display translation-invariance. For instance when presented with the goal of identifying a feline subject inside an image, the exact geographical coordinates of this subject become less crucial. On the other hand, CNNs demonstrate exceptional proficiency at detecting primary visual attributes such as edges, corners, and textures, irrespective of their spatial arrangement within the image.
Furthermore, CNNs make use of local receptive fields, ensuring that each neuron establishes connections only with a limited region of the input data. This design choice enables CNNs to focus on local patterns, rendering them less susceptible to global changes within the input—a feature notably absent in fully connected networks.
Importantly, CNNs are very resistant to overfitting because they share parameters and have local receptive fields. This is especially true when working with large datasets. Fully

connected networks, with their profusion of parameters, tend to be more susceptible to fitting noise into the data.
The computational efficiency of CNNs emerges from their judicious use of fewer parameters and their aptitude to harness the inherent structure of the data they process. This efficiency proves invaluable, particularly when confronting high-dimensional data such as images and videos.

2.7.1 Convolution layer

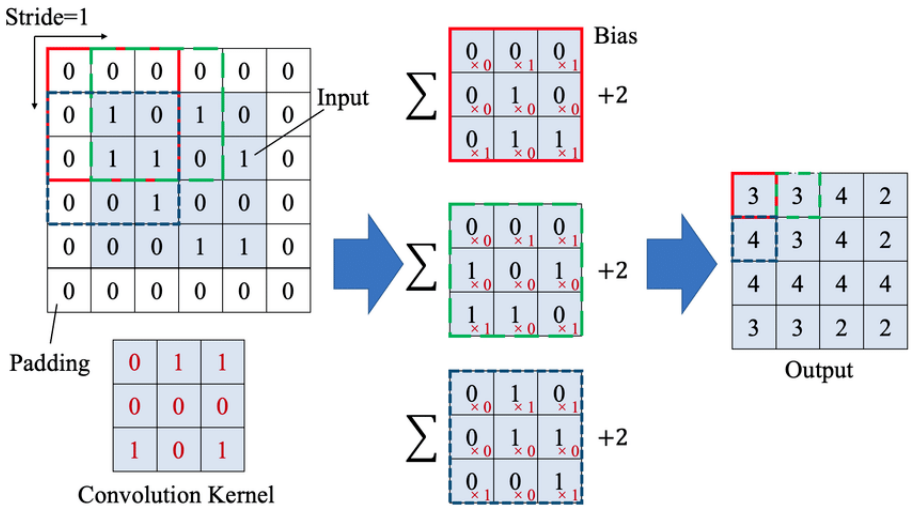

Figure 2.13: Convolution operation illustration.

Convolutional neural networks are specialized networks for grid-like topologies like images. The term convolution is inspired by the convolution operation in engineering or pure mathematical fields. It is the most commonly used term in the field of machine learning and is much like the true convolution operation, with the exception that the kernel or filter is not flipped. The accurate term for the convolution operation performed by many deep learning frameworks is cross-correlation. The reason behind this is to eliminate an inconsequential operation and reduce computational costs. The network's adaptative nature allows it to learn the appropriate filters, regardless of whether they are flipped or not.
The operation of the convolution layer is described as follows:
Initially, the kernel is positioned at the top-left corner of the input data and subsequently slides across the input data, one step at a time. At each step, an element-wise multiplication is performed between the values of the kernel and the receptive field, which are the corresponding values of the input data that are currently overlapping with the kernel. After the element-wise multiplication, the products are summed to obtain a single value. This value is placed in a new matrix known as the feature map. The feature map represents the result of the convolution operation for that particular position of the kernel. The kernel continues to slide across the input data, repeating the multiplication and summation

processes at each step. When this iterative process covers the entire input data, the result is a feature map that captures how the input data and kernel interact at various positions, representing the responses of the kernel to different patterns or features within the input data. The rate at which the kernel moves is known as the stride, and it can be greater than 1. A larger stride reduces the spatial dimensions of the feature map. Padding can be added to the input data to prevent the feature map dimensions from shrinking. Zero padding is the process of adding extra rows and columns of zeros around the input data before convolution to prevent information loss at the edges and preserve the spatial dimensions. In CNNs, multiple kernels with different learned weights are used to scan the input data simultaneously. Each kernel is sensitive to a specific feature, such as edges, textures, or shapes. Subsequent layers then process the resulting feature maps to learn complex hierarchical features for a variety of tasks, including object detection and image classification.

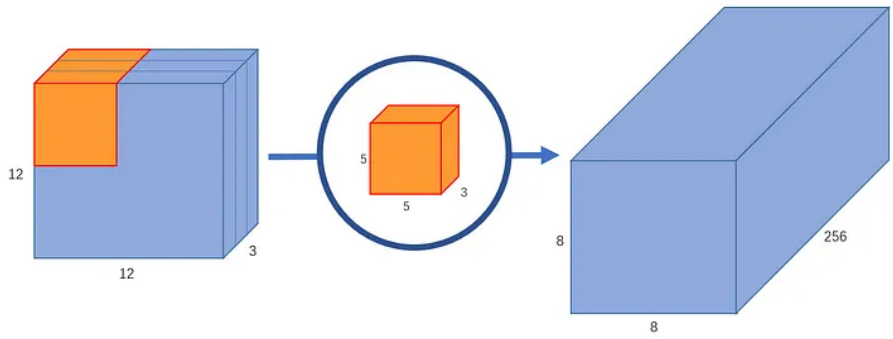

Figure 2.14: 5x5 Convolution applied to a 12x12x3 image with no padding.

2.7.2 Depthwise Convolution

Depthwise separable convolutions divide a kernel into two separate parts: the depthwise convolution and the pointwise convolution.

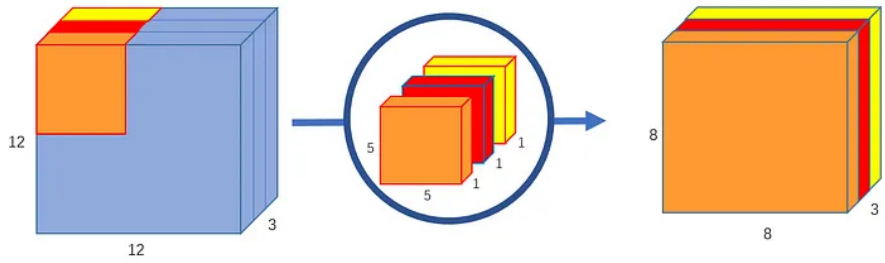

Figure 2.15: Depthwise Separable Convolution.

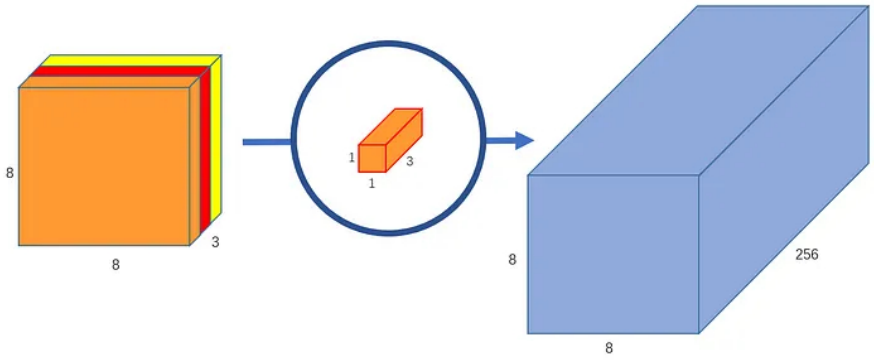

Figure 2.16: 1x1 (pointwise) convolution applied afterwards with 256 filters.

The depthwise separable convolution processes the input without altering the depth (number of channels) and employs smaller kernels for each channel. This results in multiple channel interpretations, which are then combined using the pointwise convolution, employing 1x1 kernels, to produce the final output.

The key advantage of depthwise separable convolutions is their computational efficiency. Traditional convolutions involve a significant number of multiplications due to multiple transformations of the image. In contrast, depthwise separable convolutions transform the image only once in the depthwise convolution and subsequently expand it in the pointwise convolution, resulting in significantly fewer multiplications. This reduction in computations enhances the network's efficiency and speed.

However, depthwise separable convolutions are not without limitations. When used in small networks, they can lead to a shortage of parameters, hindering effective learning during training. Nevertheless, when applied appropriately, they strike a balance between efficiency and effectiveness, making them a popular choice in modern CNN architectures, particularly in resource-constrained scenarios.

Additionally, 1x1 kernels, which are integral to pointwise convolutions, have broader applications beyond separable convolutions. They are employed to adjust the depth of an image and introduce non-linearity, enhancing the network's capacity for non-linear transformations, which is a crucial aspect of deep learning models.

2.7.3 Pooling layer

Pooling layers are used to downsample feature maps. As a consequence of reducing their size, the number of parameters and computation costs are also reduced.

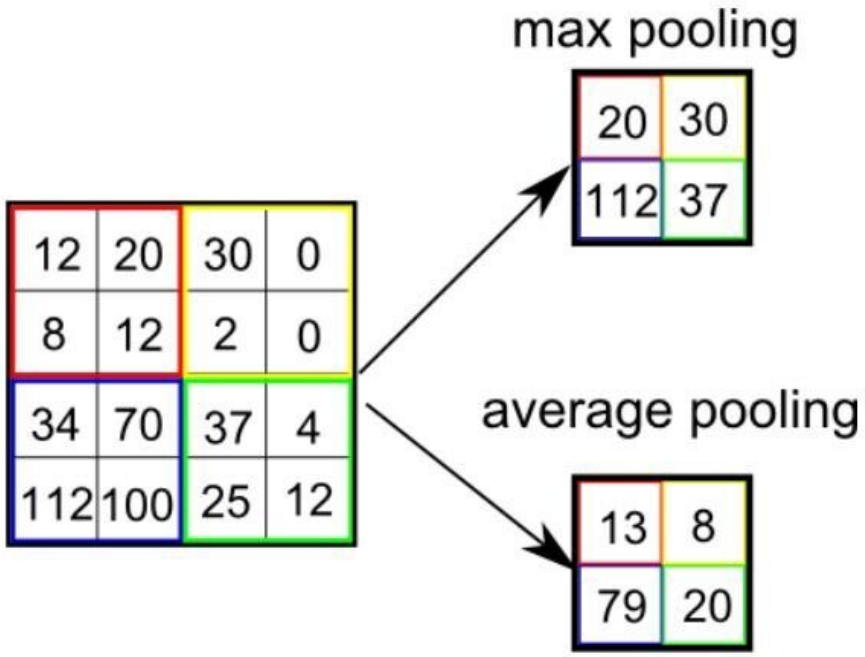

Figure 2.17: Visual explanation of max pooling and average pooling.

In max pooling, the maximum value within a region is retained. This allows the network to focus on the most important features while discarding less important details.

In average pooling, the average value of the region is retained, and information is not discarded. While it accomplishes the goal of downsampling the feature map, the most important features are diluted with possibly irrelevant information.

2.8 Hyperparameter tuning

During training, the machine learning model tunes parameters such as the network's weights and biases in order to find the best possible configuration. However, it is the researcher's responsibility to set up hyperparameters that aren't learned during training but significantly impact the model's performance, like learning rate, batch size, number of layers, number of neurons per layer, dropout rate, regularization strength, etc. Hyperparameter tuning is the process of searching for the optimal combination of hyperparameters that results in the best performance on the given dataset, for which there are no fixed rules and thus usually involves a trial-and-error process.

2.8.1 Manual search

Manual search involves using well known combinations of hyperparameters and making adjustments based on the knowledge and experience of the researcher until the results are satisfactory. This can be time-consuming and may not fully explore the hyperparameter space.

2.8.2 Grid search

Grid search is a method of systematically training and evaluating the model according to a grid of every possible combination of specified hyperparameter values. This can be computationally expensive, and the specified hyperparameter values may not be the most optimal.

2.8.3 Random search

Random search performs random samples of hyperparameters from specified ranges of values. This method is more efficient than grid search and performs well even in high-dimensional spaces.

2.8.4 Bayesian optimization

Bayesian optimization builds a probabilistic model of the objective function and uses this model to guide the search for the best hyperparameters efficiently, thus reducing the number of evaluations needed. It's particularly useful in scenarios where the objective function is expensive to evaluate and might have noisy or uncertain outcomes.

Chapter 3

State-of-the-Art of facial expression recognition

This section focuses on the most important recent advancements in facial expression recognition using deep learning methods. This section also explores different approaches, architectures, datasets, and enhancement techniques employed in facial expression recognition models. Early facial expression recognition relied on handcrafted feature extraction processes like Gabor filters, HoG, and Haar-like features to classify facial expressions. However, the emergence of deep learning techniques has managed to overcome many limitations of these methods, allowing them to capture more complex patterns and variations.

3.1 ImageNet Large Scale Visual Recognition Challenge

Many deep learning architectures in the field of computer vision rose to prominence in the ImageNet Large Scale Visual Recognition Challenge (ILSVRC) [11].

ILSVRC, also simply known as ImageNet, was an annual computer vision competition that aimed to advance the field of image classification and object recognition. It played a significant role in driving progress in deep learning and convolutional neural networks (CNNs) and served as a benchmark for evaluating the performance of various computer vision models. ImageNet is primarily associated with its large-scale dataset [11]. The dataset contains millions of labeled images belonging to thousands of different classes or categories. Initially, the dataset contained around 1.2 million images in 1,000 categories, but it has since evolved with more images and classes in later versions. Participants were required to train models to correctly classify images into one of the predefined categories. In later years, additional tasks were introduced, such as object detection and localization, where models had to not only classify objects but also identify their positions within images. For image classification, the primary evaluation metric used was ”top-1 accuracy,” which measures the percentage of test images correctly classified into their true categories. ”Top-5 accuracy” was another metric used, which measures the percentage of test images where the correct category was among the top 5 predicted categories. ImageNet provided a standardized benchmark that allowed researchers to compare the performance of different image classification models objectively. It also spurred innovation in deep learning, leading to the development of more powerful CNN architectures, such as AlexNet, VGG, GoogLeNet, ResNet, MobileNet, EfficientNet and more.

3.2 Metrics

Evaluating the performance of deep learning models is a critical aspect of model assessment. Various metrics are employed to measure different aspects of a model's performance, depending on the specific task and objectives. Some of the key metrics used for assessing deep learning models include:

3.2.1 Accuracy

Accuracy is a fundamental metric that measures the proportion of correctly classified instances among all instances in a dataset. It provides an overall view of the model's correctness but may not be suitable for imbalanced datasets.
Accuracy is calculated as follows:

$$\text{Accuracy} = \frac{\text{True Positives + True Negatives}}{\text{True Positives + True Negatives + False Positives + False Negatives}} \tag{12}$$

3.2.2 Precision

Precision measures the proportion of true positive predictions out of all positive predictions made by the model. It focuses on the accuracy of positive predictions and is particularly useful when minimizing false positives is crucial.
Precision is calculated as follows:

$$\text{Precision} = \frac{\text{True Positives}}{\text{True Positives + False Positives}} \tag{13}$$

3.2.3 True Positive Rate

True Positive Rate, also named Recall Sensitivity, measures the proportion of true positive predictions out of all actual positive instances. It emphasizes the model's ability to identify all relevant instances and is crucial in scenarios where missing positive cases is costly.
Recall is calculated as follows:

$$\text{Recall} = \frac{\text{True Positives}}{\text{True Positives + False Negatives}} \tag{14}$$

3.2.4 True Negative Rate

True Negative Rate or specificity, measures the proportion of true negative predictions out of all actual negative instances. It is essential when correctly identifying negatives is crucial, such as in medical diagnostics.
True Negative Rate is calculated as follows:

$$\text{True Negative Rate} = \frac{\text{True Negatives}}{\text{True Negatives + False Positives}} \tag{15}$$

3.2.5 F1 Score

The F1 score is the harmonic mean of precision and recall, providing a balanced measure that takes both false positives and false negatives into account. It is particularly useful when you want to strike a balance between precision and recall.
The F1 score is calculated as the harmonic mean of precision and recall:

$$\text{F1 Score} = \frac{2 \cdot \text{Precision} \cdot \text{Recall}}{\text{Precision + Recall}} \tag{12}$$

These metrics, among others, offer valuable insights into a deep learning model's performance and are chosen based on the specific goals and characteristics of the problem at hand. It is essential to select and interpret these metrics thoughtfully to gain a comprehensive understanding of a model's strengths and weaknesses in various real-world applications.

3.3 State-of-the-art methods for FER

3.3.1 Pre-training and fine-tuning

Pre-training and fine-tuning are essential techniques for addressing overfitting in deep facial expression recognition (FER) networks due to limited data. Researchers use additional task-specific data to either pre-train their networks from scratch or fine-tune existing well-known models like those that emerged from the ILSVRC. The use of such additional data, often from large-scale face recognition (FR) datasets or relatively large FER datasets, helps increase the model's capacity without overfitting, improving FER performance. For instance, VGG-Face, originally trained for FR, outperformed ImageNet, developed for object recognition. Studies have shown that pre-training on larger FR datasets positively impacts emotion recognition accuracy, and further fine-tuning with FER datasets enhances performance. Instead of directly using pre-trained models for feature extraction, a multistage fine-tuning strategy is employed for better results. This strategy involves an initial fine-tuning using FER datasets, followed by a second-stage fine-tuning on the target dataset, making the model more specific. However, pre-training and fine-tuning on external FR data may leave face-related features in the learned model, potentially weakening its ability to represent expressions. To address this, a two-stage training algorithm called FaceNet2ExpNet was proposed. It utilizes a fine-tuned face net as initialization for the expression net and guides the learning of convolutional layers. Fully connected layers are then trained from scratch with expression information, ensuring better representation of facial expressions.

3.3.2 Network Emsembles

This approach involves amalgamating multiple networks, a strategy that has demonstrated superior performance compared to individual networks. Two critical aspects of effectively

implementing network ensembles are diversity and the selection of an appropriate ensemble method. To ensure complementarity among the constituent networks, a multifaceted approach is adopted, encompassing various elements. This includes the incorporation of diverse training data, achieved through the use of diverse datasets or data generation techniques such as deformations, normalizations, and feature variations. Furthermore, network diversity is enhanced by varying network parameters, which entails modifying filter sizes, neuron quantities, layer counts, and weight initialization seeds. Additionally, diversification is achieved through the use of various network architectures, encompassing both supervised and unsupervised models. In the ensemble framework, ensemble members are fused at two levels: the feature-level and the decision-level. Feature-level ensembles involve concatenating features learned from different networks into a single feature vector. Decision-level ensembles combine network outputs using rules like simple average, weighted average, or majority voting. Weighted average methods take into account how important and reliable each network is. Techniques for determining these weights include random search, log-likelihood loss, hinge loss, exponentially weighted average, and learned weights through a CNN. These ensemble techniques collectively aim to leverage the diversity of networks and their collective decision-making to enhance FER performance.

3.4 Relevant Convolutional Neural Network architectural elements

Early CNN architectures were predominantly constructed with a repetitive pattern of convolutional layers interleaved with pooling layers. These pooling layers, while effective in downsampling feature maps, would often discard valuable spatial details in the process.

3.4.1 Strided Convolution

The adoption of strided convolutions as an alternative to pooling operations marked a transformative shift [12].
Strided convolutions offered an alternative method to downsampling while preserving more spatial information when compared to pooling operations. This preservation is achieved by applying a convolutional operation that considers the contextual relationship between neighboring pixels. Strided convolutions also have a smooth and differentiable downsampling operation, which is a key part of backpropagation and gradient computation that works well. In contrast, the max pooling operation in particular introduced non-differentiable operations, posing challenges during training.

3.4.1.1 Residual connection

The incorporation of residual blocks in Residual Neural Networks (ResNet) [13]introduced a significant advancement. These blocks featured skip connections or shortcut connec-

tions, which were inspired by the concept of residual learning, to avoid intermediary levels. The primary objective of this architectural innovation was to acquire knowledge about residual functions in relation to identity mapping. In the context of a residual block, the input feature map underwent a series of convolutional layers, and the resulting output from these layers was combined with the original input using an element-wise addition operation. The inclusion of this addition operation played a crucial role in the preservation of gradient flow throughout the training process. The inclusion of skip connections in the network enables the network to effectively understand the discrepancy between the desired output and the input feature map, hence greatly simplifying the training process for networks with a large number of layers. This characteristic proved to be highly beneficial in effectively mitigating the vanishing gradient problem, hence facilitating the successful training of neural networks comprising numerous layers. CNNs demonstrate a notable ability to capture characteristics that display translation invariance. When presented with the goal of identifying a feline subject inside an image, the exact geographical coordinates of this subject become less crucial. In contrast, CNNs have exceptional proficiency at detecting elemental characteristics such as edges, corners, and textures, irrespective of their spatial arrangement within the image.

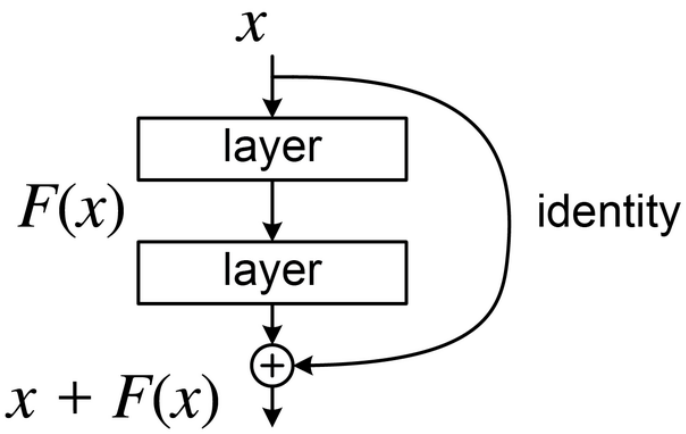

Figure 3.1: Residual connection.

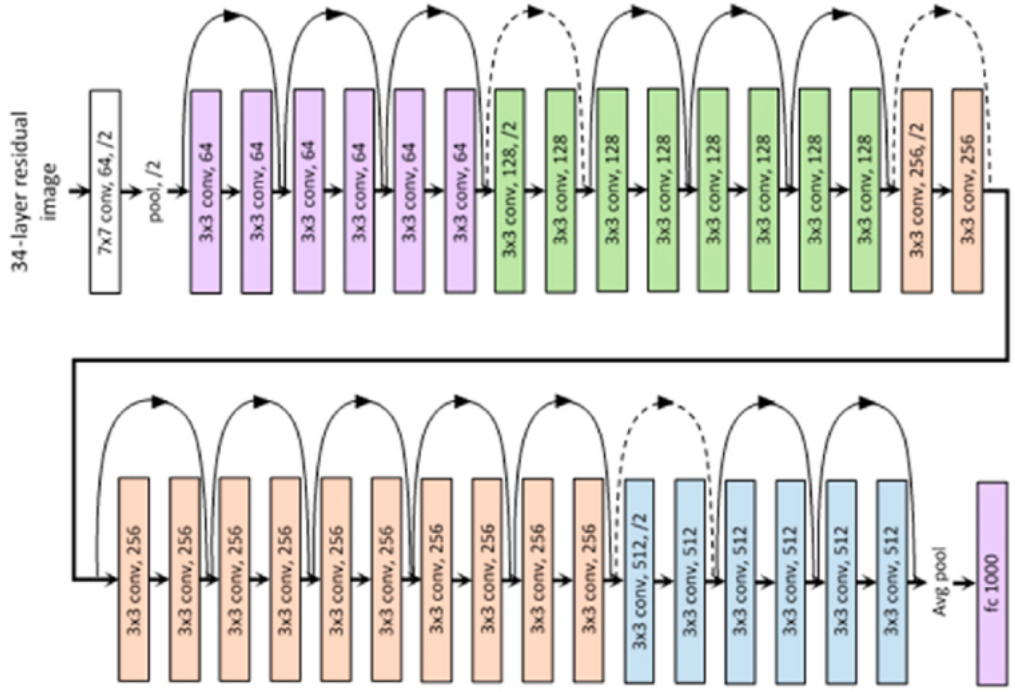

Figure 3.2: ResNet34 architecture diagram.

3.4.2 MBConv bottleneck structure

MobileNetV2 [14] introduced the concept of Inverted Residual Blocks, also known as MBConv blocks, to enhance computational efficiency while preserving model performance. In contrast to traditional Residual Blocks with a wide-narrow-wide structure of channel numbers, Inverted Residual Blocks featured a narrow-wide-narrow bottleneck configuration. This design consisted of a pointwise (1x1) convolution for channel reduction, followed by a depthwise separable convolution to capture spatial information, and culminated in another pointwise convolution to expand channel dimensions. The final output was then fused with the input via element-wise addition. This novel approach capitalized on 1x1 convolutions for both channel expansion and reduction, effectively capturing intricate patterns with fewer parameters and thus enhancing computational efficiency.

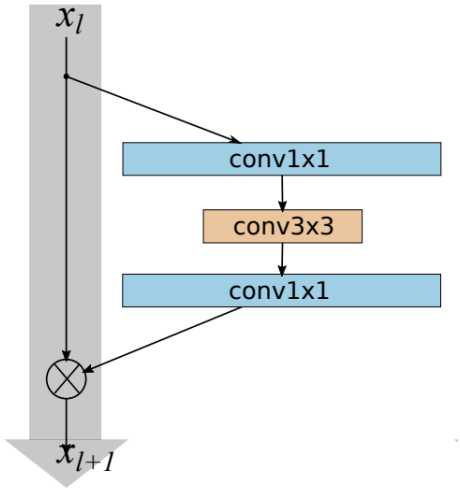

Figure 3.3: MBConv or Inverted Residual block.

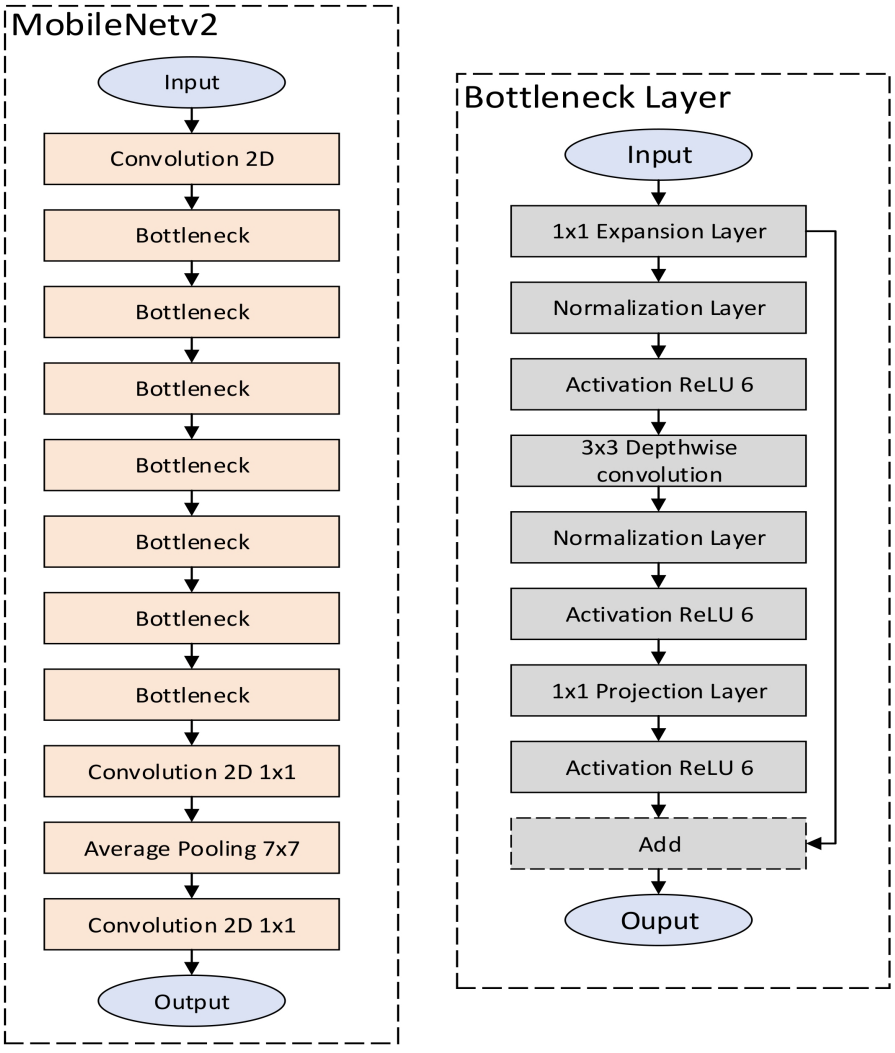

Figure 3.4: MobileNetV2 architecture diagram.

3.5 EfficientNet

EfficientNet [15, 16] is a family of deep CNNs that revolutionizes the landscape of computer vision tasks by offering a highly efficient and scalable architecture.

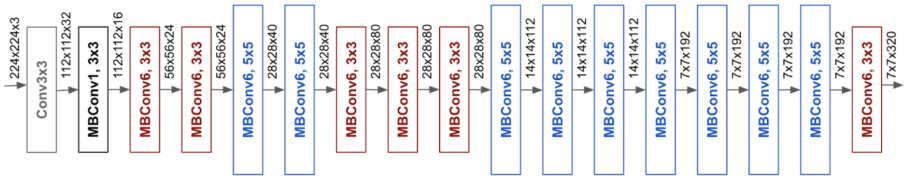

Figure 3.5: EfficientNet architecture diagram.

This innovation is achieved through the introduction of compound scaling, which entails the uniform scaling of network width, depth, and resolution.

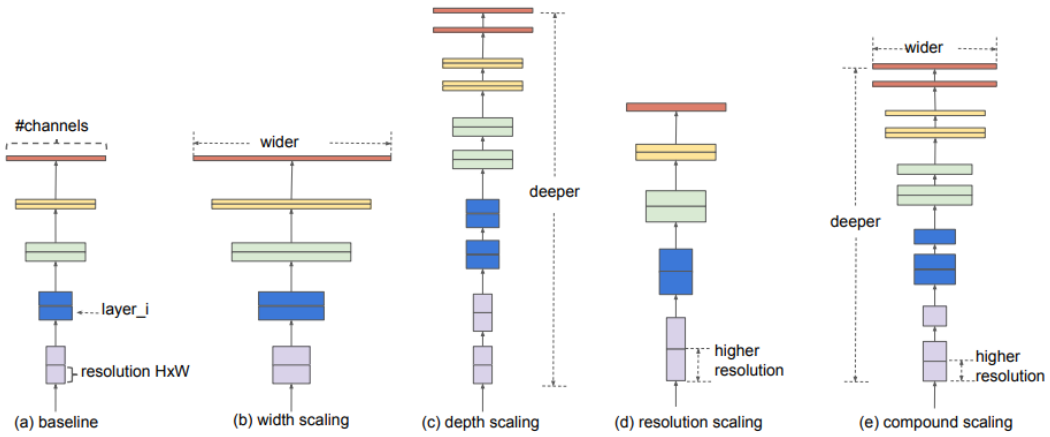

Figure 3.6: EfficientNet scaling parameters.

This is the structure of EfficientNet. It was made through a neural architecture search (NAS) [17] process that aimed to find the best mix of building blocks and scaling factors. NAS helps identify the most effective architectural choices and facilitates the selection of architectural choices that strike a balance between model complexity and performance within given constraints, a pivotal aspect of optimizing efficiency. Additionally, EfficientNet incorporates skip or shortcut connections akin to ResNet, addressing the vanishing gradient problem and facilitating the training of deep networks. EfficientNet also uses inverted residual blocks, which are similar to MobileNetV2. It also uses depth-wise separable convolutions and bottleneck structures to cut down on computational overhead while keeping the power of representation.

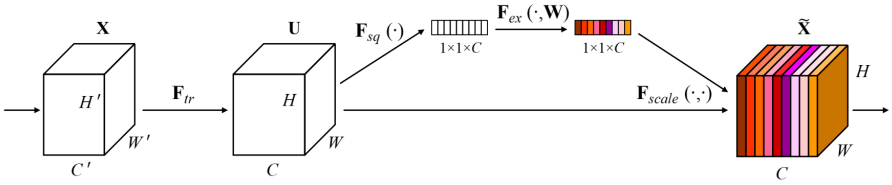

Figure 3.7: Squeeze-and-Excitation block.

In addition, Squeeze-and-Excitation (SE) Blocks, introduced by the SENet [18], are integrated into EfficientNet for channel recalibration. These blocks help the network focus on more informative channels, improving its discriminative power. Additional enhancements, including strided convolutions and the use of the Swish activation function instead of ReLU, contribute to the network's efficiency and discriminative power. EfficientNet has consistently demonstrated its efficacy across diverse benchmark datasets, like the ILSVRC [11] achieving competitive or state-of-the-art results while significantly reducing parameter count and computational resource requirements compared to larger-scale models.

3.6 Recent breakthroughs and trends

Currently, the development of learning-based approaches to FER mostly focuses on overcoming challenges caused by the diversity of training data and non-expression-related factors like illumination, head orientation, and individual features [19]. Pre-training and fine-tuning are frequently used as approaches to reduce the problems caused by a lack of data and the overfitting phenomenon. It has been found to be quite successful to use multi-stage pre-training, which involves initial training on generic data and final fine-tuning on specialized FER data. However, publicly accessible pre-trained models on the market continue to include traits that are unrelated to expressions, which can impair their performance. The development of competitive training algorithms that cover the full process while utilizing deep networks of moderate size has been aided by the availability of large and diverse facial expression recognition datasets gathered in real-world contexts.
To improve network robustness, several pre-designed features and a variety of input data types are advised, but they could also raise computing complexity. It is effective to build deep, wide networks with numerous layers and adaptable filters, but this takes a large amount of training data. Although it incurs a computational cost, integrating smaller networks in parallel or series can help solve this problem. Multitask networks can jointly train for FER and related tasks, untangling components that are unrelated to expression, but this requires labeled data for every task and is more difficult as the number of tasks increases. Although end-to-end training is preferable for higher performance, cascaded networks successively train, gradually untangling extraneous components.
Current FER networks frequently don't test in naturalistic circumstances or specifically handle differences in head poses. By frontalizing facial photos, synthesizing multiple po-

sitions, and detaching identity variations, generative adversarial networks (GANs) can be useful. They also help to improve training data. GAN training, though, can be unpredictable.

Chapter 4

Methodology

4.1 Research Hypothesis and Investigation Focus

In contemporary research, a prevailing approach to the domain of facial expression recognition (FER) involves the utilization of pre-trained networks on ImageNet, followed by the application of transfer learning. This strategy, undoubtedly effective, not only expedites the process by obviating the necessity of training a network from scratch but also exploits the rich feature representations learned by the pre-trained model.
However, it is worth noting that within the ImageNet dataset, a significant proportion—specifically, 997 out of 1000 categories in the ILSVRC—are not directly pertinent to human subjects. This surplus of non-human-related categories introduces a substantial amount of extraneous information for the specific task of FER. Given this observation, it is pertinent to contemplate whether a specialized network designed explicitly for FER might yield comparable or even superior performance while employing fewer parameters. This hypothesis prompts further exploration, consequently establishing itself as one of the objectives of this research.

4.2 Datasets

Facial expression datasets played a crucial role in the development of Facial Expression Recognition (FER) systems and Convolutional Neural Networks (CNNs) designed for this task by providing the necessary data for training and evaluation, driving the development of more effective model architectures, and enabling the application of deep learning techniques to real-world problems related to emotion recognition and human-computer interaction. These datasets have significantly influenced the fields of computer vision and deep learning in several ways: Facial expression datasets provide large amounts of labeled training data that are essential for training and fine-tuning CNN models. These datasets consist of images or videos with annotated facial expressions, allowing researchers and developers to create accurate and robust FER models. Datasets provide a benchmark for evaluating the performance of FER systems and CNN architectures. Researchers are able to compare the accuracy of different models on standardized datasets, enabling fair comparisons and driving innovation. The availability of facial expression datasets has driven innovation in CNN architectures specifically tailored for FER. Researchers have designed architectures with specialized layers and structures to better capture facial features and expressions, improving model performance. Facial expression datasets have facilitated the development of FER systems for a wide range of real-world applications, including

emotion analysis in human-computer interaction, sentiment analysis in marketing, and mental health monitoring through facial expression analysis.
Facial expression datasets come from diverse sources and capture a wide range of expressions under different conditions. This diversity helps FER models generalize better to real-world scenarios, making them more robust in recognizing facial expressions in various lighting conditions, poses, and backgrounds.

4.2.1 FER2013

The FER2013 dataset [20] is a widely used dataset in the field of Facial Expression Recognition (FER). It is a benchmark dataset designed for training and evaluating FER systems.

Figure 4.1: Example images from FER2013 with corresponding emotions.

The FER2013 dataset encompasses a comprehensive collection of 35,887 grayscale images, all uniformly sized at 48x48 pixels. These images are partitioned into two distinct subsets: the training set, containing 28,709 images, and the validation set, containing 7,178 images. The FER2013 dataset consists of seven emotion categories. The distribution of these is included in the following table.

Table 4.1: Emotions and number of samples in training and validation sets.

Emotion	Training	Validation
Angry	3995	958
Disgust	436	111
Fear	4097	1024
Happy	7215	1774
Neutral	4965	1233
Sad	4830	1247
Surprise	3171	831

The images in the FER2013 dataset were collected from various sources, including the internet. The data collection process involved using search engines to find images with relevant keywords associated with each emotion class. After that, human annotators manually filtered and labeled the images to assign the appropriate emotion labels. The distribution of images across emotion classes in the FER2013 dataset is not balanced, with some classes having more samples than others, which may affect model training and evaluation, as models may perform better on emotions with more training samples. Researchers and

developers use the FER2013 dataset for training and evaluating FER models, especially those based on CNN architectures. It serves as a benchmark dataset for testing the performance of FER systems in recognizing basic facial expressions. It has been used in numerous academic papers and competitions related to emotion recognition and computer vision. Despite its widespread use, the FER2013 dataset has some limitations, including the fact that the images are relatively low-resolution and may not fully capture the complexity of real-world facial expressions. The dataset's labels are also not always precise, as they are based on human annotation, which can be subjective. Some researchers have extended or combined the FER2013 dataset with other datasets to create larger and more diverse datasets for improved model training and evaluation. Variants of the FER2013 dataset may exist with additional preprocessing or augmentations to address its limitations. In summary, the FER2013 dataset is a widely recognized resource for the development and evaluation of FER systems. While it has some limitations, it has been instrumental in advancing research in emotion recognition using machine learning and deep learning techniques. Researchers often use it as a starting point and may build upon it with additional data or improvements to address its shortcomings.

4.2.2 AffectNet

The AffectNet dataset [21] is a comprehensive and widely used dataset in the fields of affective computing and facial expression recognition (FER). It was created to facilitate research and development in the area of emotion recognition by providing a large and diverse collection of facial images annotated with emotional labels.

Figure 4.2: Example images from AffectNet with corresponding emotions and valence and arousal values.

AffectNet contains over 1 million facial RGB images with varying resolutions. It is divided into three subsets: The training set contains approximately 450,000 images. The vali-

dation set contains around 56,000 images. The test set contains approximately 56,000 images. AffectNet provides labels for eight emotion categories, representing a broader range of emotions than some other datasets. The distribution of these is included in the following table.

Table 4.2: Emotions and number of samples in training and validation sets.

Emotion	Training	Validation
Angry	24882	500
Disgust	3803	500
Fear	6378	500
Happy	134415	500
Neutral	74874	500
Sad	25459	500
Surprise	14090	500
Contempt	3750	499

In addition, AffectNet includes annotations using the Valence-Arousal scales from the Circumplex dimensional model of affect [2], which holds potential significance for models designed to classify emotions within a continuous spectrum as opposed to rigidly discrete categories. AffectNet images were collected from various sources, including the internet and social media platforms. The dataset creators used automatic facial expression analysis software to select and filter images based on the presence of facial expressions. Human annotators were employed to assign emotional labels to the selected images. The distribution of images across emotion classes in AffectNet is relatively balanced compared to some other datasets, which can be beneficial for training and evaluating FER models. The dataset aims to capture a wide range of emotional expressions in diverse real-world scenarios. AffectNet is widely used in the research and development of FER systems, machine learning models, and deep learning architectures. It serves as a valuable resource for training and evaluating models that aim to recognize a broader spectrum of emotions beyond the basic emotional states. Despite its advantages, the AffectNet dataset also has some challenges, including potential noise in the data due to its diverse sources and annotation subjectivity. The dataset may contain variations in lighting, pose, and image quality, reflecting the real-world conditions from which the images were collected. Researchers have extended or combined the AffectNet dataset with other datasets to create even larger and more diverse datasets for improved model training and evaluation. Variants of AffectNet may exist with additional preprocessing, augmentations, or refined annotations. In summary, the AffectNet dataset is a significant resource for the development of FER systems and emotion recognition research. Its large size, diverse emotions, and real-world nature make it valuable for training and testing models that aim to recognize a wide range of human emotional expressions. Researchers often make use of AffectNet in their work and continue to explore ways to improve FER models using this dataset.

4.2.3 Class Imbalance

Imbalanced datasets are characterized by a disproportionate representation of classes. In such datasets, one or more classes are typically referred to as the majority or dominant classes when they encompass a notably larger number of instances compared to one or more minority or underrepresented classes.

When training a model using an imbalanced dataset, the learning process exhibits a bias toward the predominant classes. As a consequence of having a greater abundance of instances from these majority classes, the model tends to be more proficient in classifying them accurately. However, this is counterbalanced by the inadequacy of sufficient examples pertaining to the minority classes, resulting in the model's limited capacity to discern and acquire valuable discriminative patterns necessary for accurate classification within the minority classes.

In this work, both datasets are heavily imbalanced, which might compromise the model's performance on the underrepresented classes if it does not take into consideration the class distribution/proportion or balance of classes. To address this issue, several techniques may be employed. Oversampling techniques focus on resampling or generating synthetic samples for minority classes. An example of this strategy is SMOTE and its variants. Undersampling techniques reduce the number of samples from majority classes while preserving the number of samples from minority classes. An example of this approach are Tomek links. Tomek links identify pairs of data points in proximity to each other, where one point belongs to the majority class and the other belongs to the minority class. Then the data point from the majority class in such pairs is removed to create a clearer separation between the classes.

Furthermore, it is worth noting that a synergistic approach, integrating both oversampling and undersampling techniques, can be harnessed to further enhance the efficacy of addressing imbalanced datasets. An alternative strategy is the use of a weighted loss function. This constitutes a modification of the conventional loss function employed during model training. This method consists of assigning greater weight to the minority class while attributing a correspondingly reduced weight to the majority class. These weight assignments are inversely proportional to the class frequencies. This effectively imparts a heightened penalty to misclassifications associated with the minority classes and diminished incentives for accurate classifications of the majority classes. The use of a weighted loss function offers a potential advantage as it maintains the dataset in its original form. This is particularly relevant because deep learning models are designed to excel in real-world scenarios, and the datasets used often consist of images sourced from the internet rather than controlled environments. In such cases, the class imbalance present in the data is considered intrinsic variation because of the distribution found in real-world situations. For these reasons, a weighted loss function was the strategy chosen for addressing the problem of the class imbalance.

4.2.4 Data augmentation

Data augmentation was incorporated as a training-time technique to enhance the model's ability to generalize. This approach, inspired by the methodology presented in [22] involved subjecting the images to random transformations, whose values mirror those specified in the referenced paper, during each training epoch.

These transformations consist of:

- Horizontal flipping
- Rotation within the range of -10 to 10 degrees
- Vertical and horizontal translation by a factor of 0.1
- Zooming in/out by a factor of 0.1
- Contrast adjustment by a factor of 0.1
- Brightness adjustment by a factor of 0.1

4.3 Architecture Selection

To address the objectives of this study, I selected the EfficientNetV2S architecture. Among the variants within the EfficientNetV2 family, EfficientNetV2S was chosen due to its relatively smaller size and architectural characteristics. Notably, compared to the earlier EfficientNetV1 family, the EfficientNetV2S architecture features smaller kernel sizes (3x3 instead of a combination of 5x5 and 3x3) along with a reduction in the number of stages. These distinctive architectural choices render it a promising candidate for this experiment.

Table 4.3: EfficientNetV2-S architecture. The MBConv and Fused-MBConv operators are described in figure 4.3.

Stage	Operator	Stride	#Channels	#Layers
0	Conv3x3	2	24	1
1	Fused-MBConv1, k3x3	1	2 4	2
2	Fused-MBConv1, k3x3	2	48	4
3	Fused-MBConv1, k3x3	2	64	4
4	MBConv4,k3x3,SE0.25	2	128	6
5	MBConv4,k3x3,SE0.25	2	160	9
6	MBConv4,k3x3,SE0.25	2	256	15
7	Conv 1x1 & Pooling & FC	-	1280	1

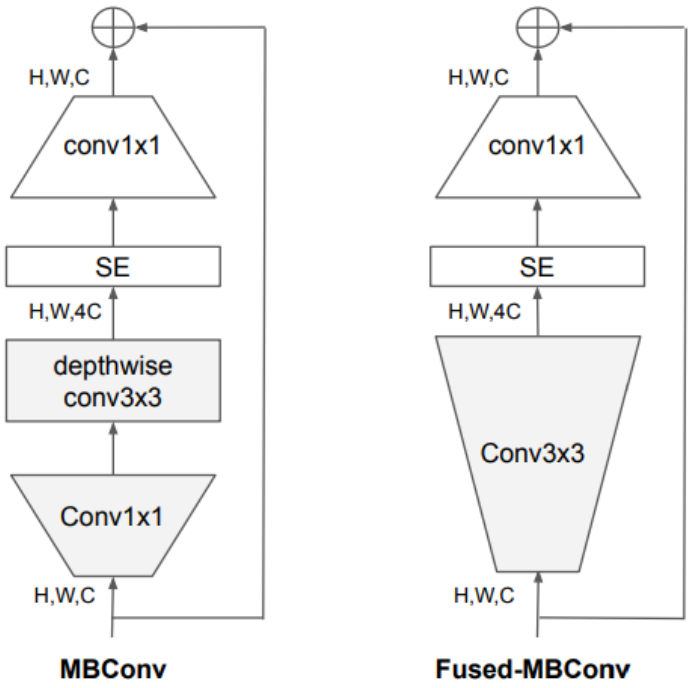

Figure 4.3: MBConv(left) and FusedMBConv(right).

4.4 Experimental Setup

4.4.1 Baseline Experiment on FER2013

In the initial experiment, the primary objective was to establish a baseline performance for the model. To achieve this, the original EfficientNetV2S architecture was retained without modification, except for two key adjustments: first, the input shape was adapted to accommodate grayscale 48x48 images (48x48x1), and second, the number of channels/filters in the final layer, initially set at 1280, was parameterized as 'numTopFilters' for subsequent optimization through Optuna. Functionally, numTopFilters serves a similar role to the number of neurons in the fully connected layer before the classification layer in traditional CNNs, a commonly used adjusted hyperparameter. The motive for the second key adjustment was to reduce the number of filters, given the rationale that the number of filters corresponds to the network's capacity to capture distinctive features for classification. In contrast to the ImageNet Large Scale Visual Recognition Challenge (ILSVRC), which encompasses 1000 classes, our task involves classifying only seven categories. Hence, it is justifiable to reduce the number of filters to optimize computational efficiency. Subsequently, the model was trained using the FER2013 dataset for 50 trials with a maximum of 100 epochs.

4.4.2 Scaled-Down Experiment on FER2013

In the second experiment, the objective was to assess the impact of complexity reduction on the models performance. Specifically, stages 5 and 6 from the EfficientNetV2S architecture were removed, and the resulting model was trained on FER2013 for 50 trials and a maximum of 100 epochs.

Table 4.4: EfficientNetV2-S architecture - MBConv and Fused-MBConv are described in figure 4.3.

Stage	Operator	Stride	#Channels	#Layers
0	Conv3x3	2	24	1
1	Fused-MBConv1, k3x3	1	2 4	2
2	Fused-MBConv1, k3x3	2	48	4
3	Fused-MBConv1, k3x3	2	64	4
4	MBConv4,k3x3,SE0.25	2	128	6
5	Conv 1x1 & Pooling & FC	-	numTopFilters	1

This modification aimed to investigate whether reducing model complexity would lead to any noticeable changes in performance.

4.4.3 Scaled-Down Experiment on AffectNet

In the third experiment, the input shape was altered in order to accept RGB images (48x48x3) and train the model on AffectNet. This allows for comparison with other works on AffectNet and examining its performance on RGB images, unlike the first experiments. In the course of training the AffectNet dataset, it is worth noting that 'contempt' was omitted from consideration as one of the target categories.

4.5 Training and Validation

4.5.1 Optimizer

In this study, the Adam optimizer was selected for several compelling reasons. Adam is known for its ability to adaptively adjust learning rates during training, making it particularly suitable for non-stationary and noisy datasets. Moreover, Adam exhibits the advantage of a reduced number of hyperparameters, alleviating the need for manual fine-tuning. This attribute contributes to a more efficient optimization process, as it mitigates the potential complexities associated with hyperparameter selection. Additionally, Adam demonstrates a tendency to converge swiftly towards favorable solutions, accelerating the overall training process.

4.5.2 Hyperparameter optimization using Optuna

Optuna [23] is a Python library for hyperparameter optimization consisting of 5 main components.

The objective function is for the machine learning model to be optimized. Hyperparameter values are input using suggest_categorical or suggest_float and it returns a scalar value to be optimized. A study represents a hyperparameter optimization run consisting of multiple trials. A trial represents a single evaluation of the objective function with a specific combination of hyperparameter values. The sampler defines how to sample the hyperparameter values at each trial. The sampler used was a TPESampler. The pruner automatically stops unpromising trials at the early stages of the training, which is equivalent to early-stopping.

4.5.2.1 Sampler

The Tree-structured Parzen Estimator (TPE) sampler [24] is a probabilistic model-based optimization technique employed for hyperparameter optimization. It belongs to the class of Bayesian optimization methods, which aim to efficiently search for optimal hyperparameters within a given search space while minimizing the number of function evaluations. TPE is particularly effective when the objective function is costly to compute, as it actively selects hyperparameters to evaluate based on its probabilistic model.
The principle behind TPE is to model the distribution of good and bad hyperparameters using two separate probability density functions (PDFs): one for the ”good” configurations and one for the ”bad” configurations. These PDFs are iteratively refined to guide the search towards promising regions of the hyperparameter space. The key steps involved in the TPE algorithm can be summarized as follows:

1. First, TPE begins with an initial random sampling of hyperparameters to create an initial dataset of evaluations.

2. The evaluations are then used to construct two PDFs, denoted as ”prior” and ”posterior.” The prior represents the initial belief about the distribution of hyperparameters, often assumed to be a Gaussian distribution. The posterior, on the other hand, models the conditional probability of hyperparameters given their performance.

3. Then, to balance exploration and exploitation, TPE employs an Expected Improvement (EI) selection strategy. EI quantifies the potential improvement in the objective function if a particular hyperparameter configuration is evaluated. The algorithm selects hyperparameters that maximize EI, which encourages the exploration of promising regions while exploiting known good areas.

4. The selected combination of hyperparameters is then evaluated using the objective function. The resulting performance metric, such as accuracy or loss, is used to update the posterior PDF.

5. The posterior PDF is updated based on the evaluated configurations, refining the understanding of which hyperparameters are likely to be good or bad. This update process employs Bayesian updating, which combines prior knowledge and newly acquired information to obtain a more accurate model.

6. The sampling, evaluation, and update steps are repeated iteratively for a specified number of iterations or until a convergence criterion is met.

TPE intelligently selects the next set of hyperparameters to evaluate, focusing more resources on configurations likely to yield better results. By iteratively refining the probability distributions, TPE efficiently narrows down the search space, ultimately converging to an optimal or near-optimal set of hyperparameters.

4.5.2.2 Pruner

The pruner determines whether a trial should be terminated early based on intermediate results in order to avoid wasting computational resources on unpromising trials. Instead, early stopping has been implemented to halt training in the event of no improvement to validation loss within 10 epochs. This makes using a pruner redundant. As such, the selected optuna pruner is 'nopruner', which, as its name suggests, never prunes trials.

4.5.2.3 Hyperparameter search space

The hyperparameter search space included the following parameters:

- Learning rate, ranging from 0.01 to 0.000005
- Width coefficient, ranging from 0.2 to 1.4 in increments of 0.05
- Depth coefficient, ranging from 0.2 to 1.4 in increments of 0.05
- Dropout rate, ranging from 0 to 0.5 in increments of 0.05
- numTopFilters, ranging from 16 to 160 in increments of 8

Chapter 5

Results

This section will showcase the results obtained in the experiments described in Chapter 4. In summary, each of the experiments seeks to answer a different research question:

- First Experiment: Does training the EfficientNetV2S model from scratch on the FER2013 dataset result in effective recognition of the seven emotions, and what is the overall performance of the model in this task?
- Second Experiment: Can removing stages 5 and 6 of the EfficientNetV2S architecture lead to improved performance by reducing overfitting while also reducing the total number of model parameters?

5.1 Comparison between Unweighted and Weighted Loss

The results reported in this section were obtained in a preliminary testing phase before the experiments described in the previous chapter. A total of 50 trials with a maximum of 60 epochs were performed using FER2013. In this experiment, the same architecture as the Scaled-Down Experiment was used, with the suppression of stages 5 and 6 of EfficientNetV2S.

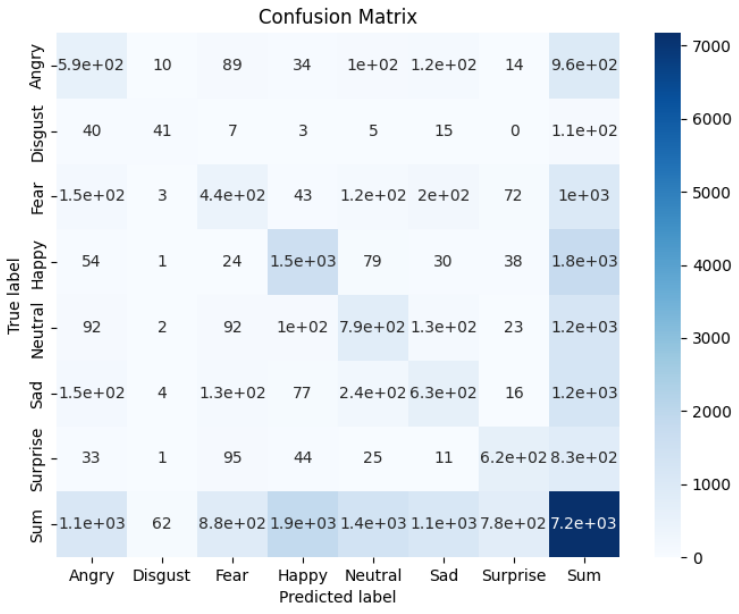

Figure 5.1: Unweighted loss Raw Confusion Matrix.

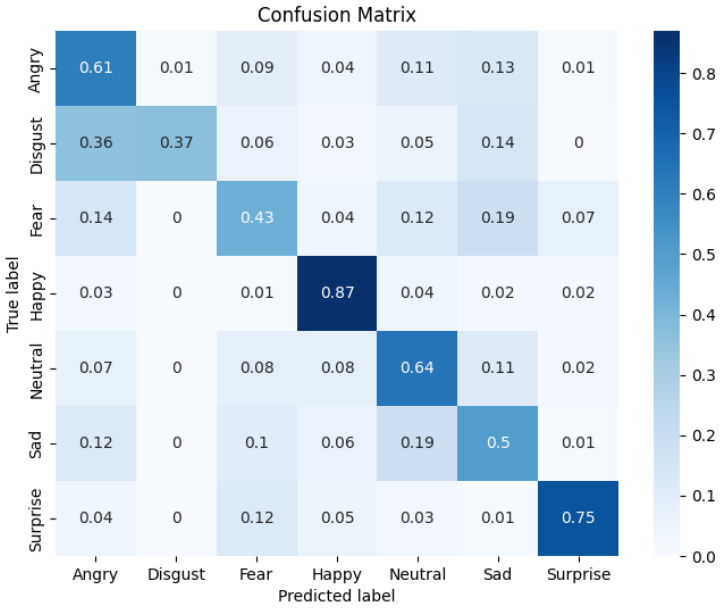

Figure 5.2: Unweighted loss Normalized Confusion Matrix.

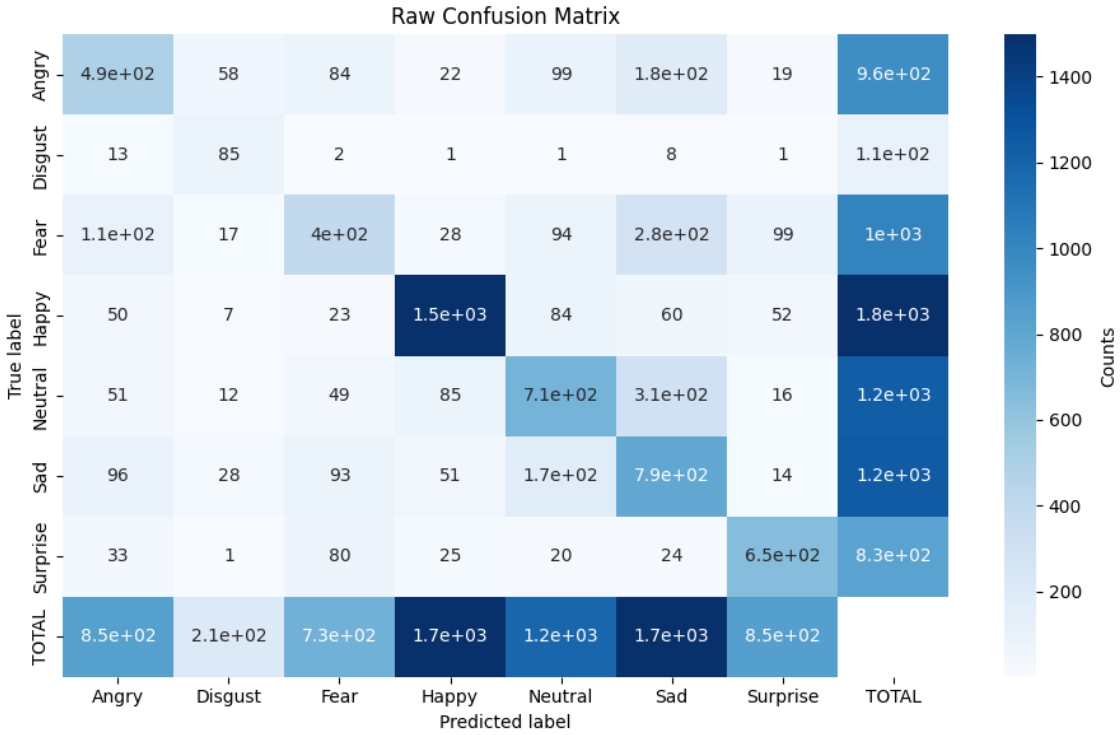

Figure 5.3: Weighted Loss Raw Confusion Matrix.

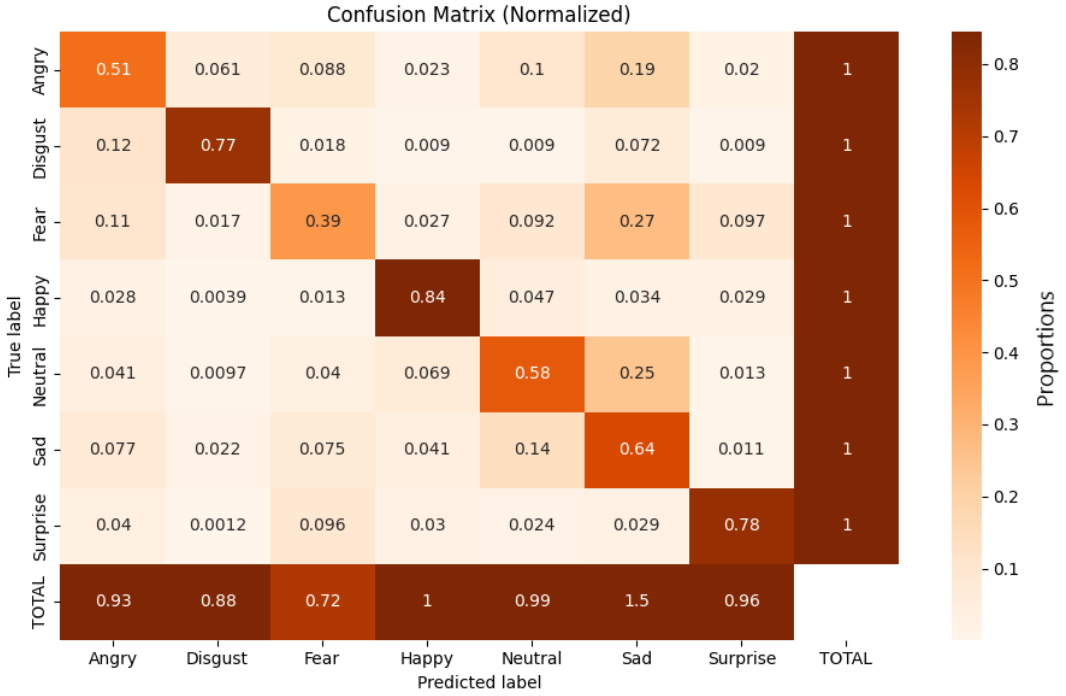

Figure 5.4: Weighted Loss Normalized Confusion Matrix.

Table 5.1: Unweighted vs Weighted Loss impact on accuracy.

	Accuracy
Unweighted	0.645
Weighted	0.644

Analysis of the confusion matrices reveals that incorporating class weights into the loss function results in the network assigning higher significance to the minority classes, ensuing in more frequent predictions on these classes, with a noticeable effect on 'disgust'. This heightened focus on minority classes is accompanied by a marginal decrease in overall accuracy. However, this reduction in accuracy is not a cause for concern, as accuracy may not be the most suitable metric for evaluating models on imbalanced datasets.

In scenarios where one class significantly dominates the dataset, a model could predict the majority class with high accuracy while failing to perform well on the minority classes. In such cases, accuracy alone does not necessarily indicate the model's effectiveness. Consequently, the evaluation of models on imbalanced datasets necessitates the inclusion of supplementary metrics such as precision, recall, and F1 score to provide a more comprehensive assessment of their performance, which will be incorporated into the next experiments.

5.2 Baseline Experiment on FER2013

Table 5.2: Performance Metrics of Baseline Experiment on FER2013.

Metric	Value
Accuracy	0.658
Loss	0.963
Precision	0.631
Recall	0.649
F1Score	0.637

Table 5.3: Best hyperparameter configuration of Baseline Experiment on FER2013.

Hyperparameter	Value
Learning rate	0.0009986
Width coefficient	0.75
Depth coefficient	1.25
Dropout rate	0.05
numTopFilters	24

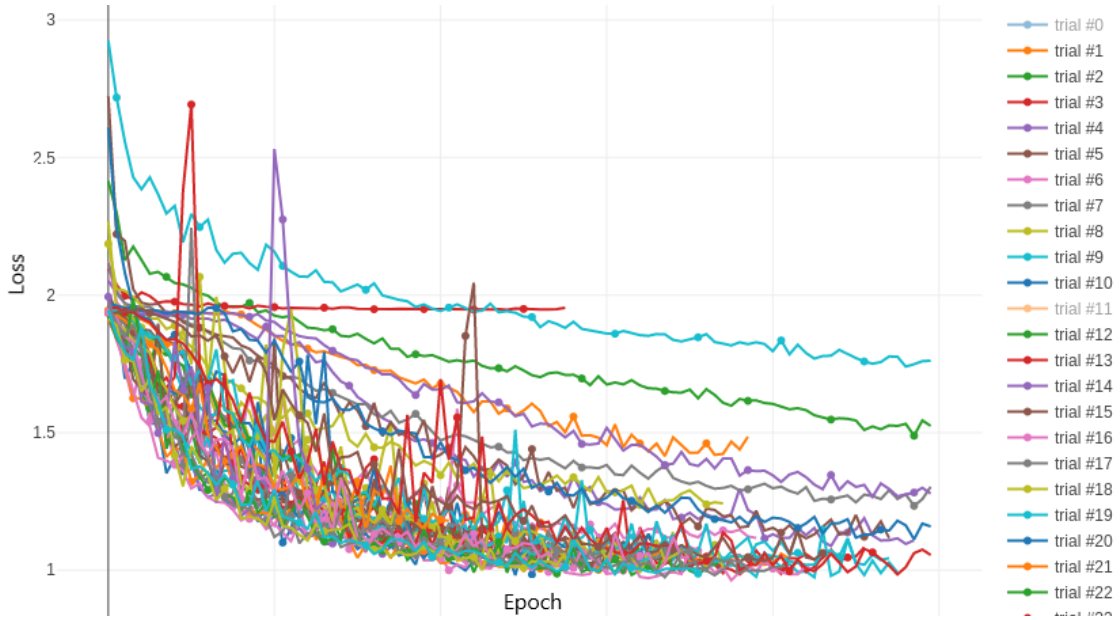

Figure 5.5: Learning curves for every trial from Baseline Experiment on FER2013.

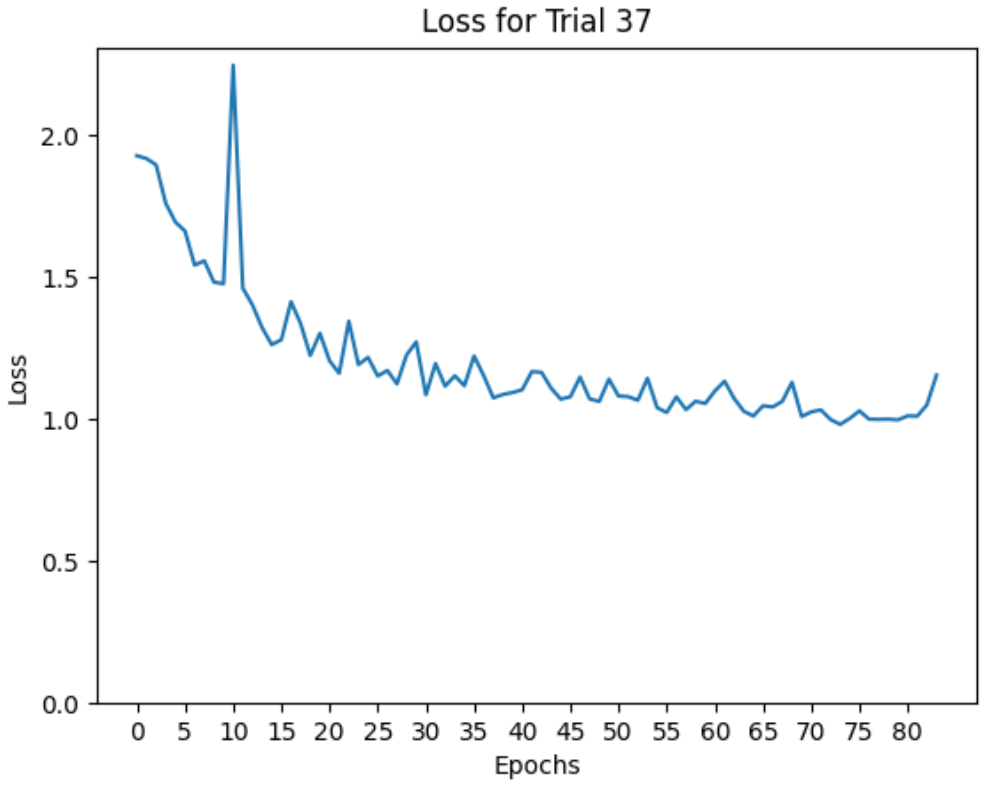

Figure 5.6: Learning curve of the best trial from Baseline Experiment on FER2013.

A learning curve is a graphical representation of the performance of a machine learning model over time or as a function of the number of training iterations or epochs. The jagged pattern observed in the curve may have arisen as a result of incorporating class weights within the loss function, which imparts a heightened penalty on the model for making erroneous predictions concerning minority classes.

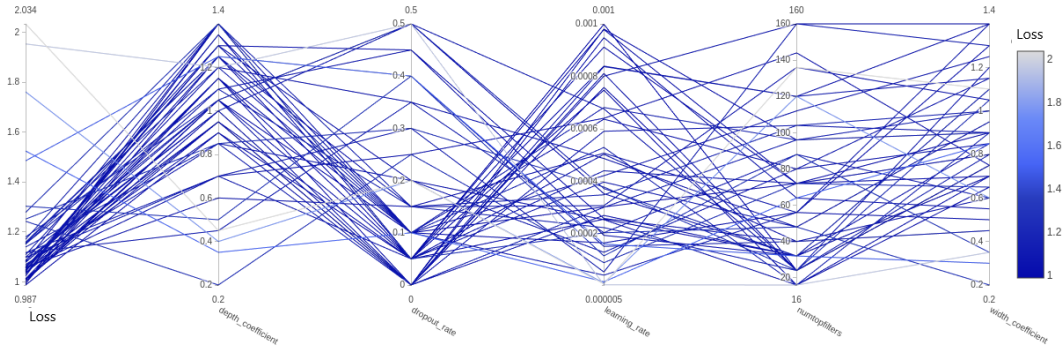

Figure 5.7: Parallel Coordinate plot from Baseline Experiment on FER2013.

Parallel coordinate plots serve as a valuable tool for the analysis of multivariate numerical data. They allow for the comparative examination of samples or observations with respect

to multiple numerical variables, with each individual feature or variable being delineated along a distinct and parallel axis.

Based on the presented graph, it can be inferred that the model exhibits a preference for a depth coefficient of approximately 1, a modest dropout rate of approximately 0.1, an intermediate learning rate in the vicinity of 0.0004, and a width coefficient of approximately 1. Notably, the variable ”numTopFilters” does not appear to demonstrate any discernible impact.

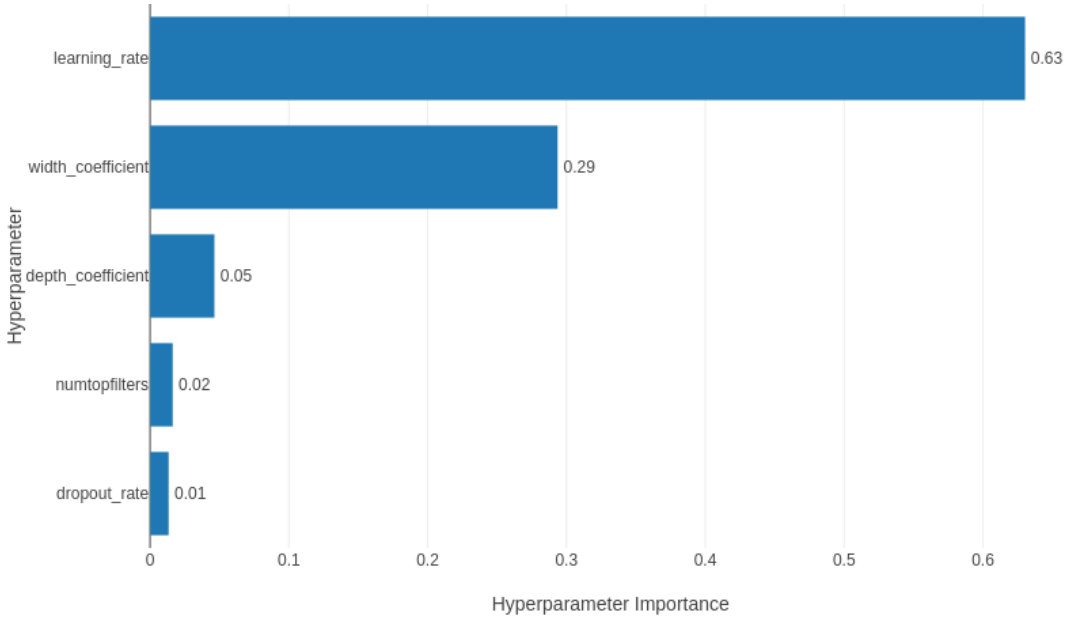

Figure 5.8: Hyperparameter Importance plot from Baseline Experiment on FER2013.

The significance of each hyperparameter’s impact on the loss function becomes evident when examining the parameter importance plot. Notably, the learning rate emerges as the most influential hyperparameter, followed closely by the width coefficient. Conversely, the remaining hyperparameters appear to exhibit comparatively minimal influence on the loss function.

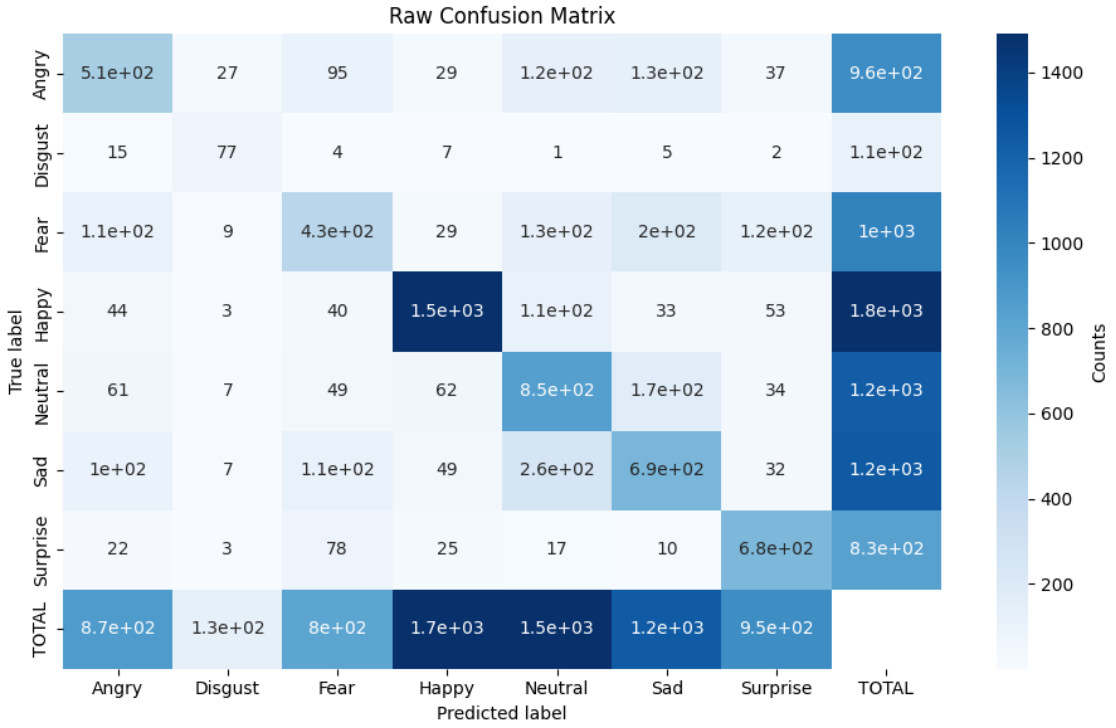

Figure 5.9: Raw Confusion Matrix for Baseline Experiment on FER2013.

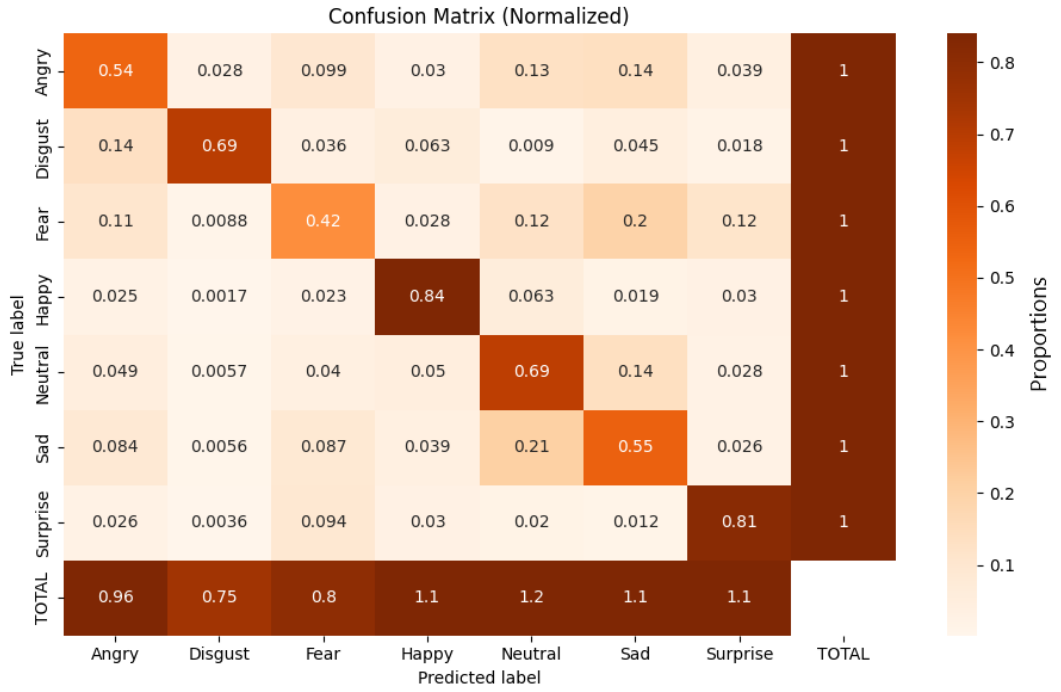

Figure 5.10: Normalized Confusion Matrix for Baseline Experiment on FER2013.

Confusion matrices evaluate the performance of a classification model by providing a clear and detailed breakdown of the model's predictions compared to the true class labels. In the confusion matrices, it is possible to observe how 'Happy' is recognized more accurately

as a result of having more training samples than the other classes. In spite of the utilization of class weights, the model continues to exhibit suboptimal performance for the minority classes. This indicates that the network may not effectively capture the nuanced distinctions between these emotional categories. It is comprehensible that discerning between 'Sad' and 'Neutral' may pose a challenge, as the associated Action Units (AUs) manifest more subtle variations.

5.3 Scaled-Down Experiment on FER2013

Table 5.4: Performance Metrics of Scaled-Down Experiment on FER2013.

Accuracy	0.657
Loss	0.959
Precision	0.637
Recall	0.649
F1Score	0.641

Table 5.5: Best hyperparameter configuration of Baseline Experiment on FER2013.

Hyperparameter	**Value**
Learning rate	0.0001972
Width coefficient	1.3
Depth coefficient	0.55
Dropout rate	0.2
numTopFilters	24

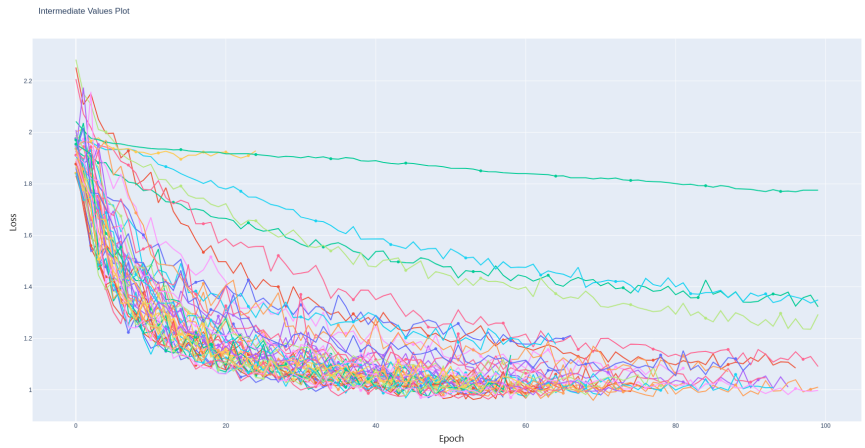

Figure 5.11: Learning curves for every trial from Scaled-Down Experiment on FER2013.

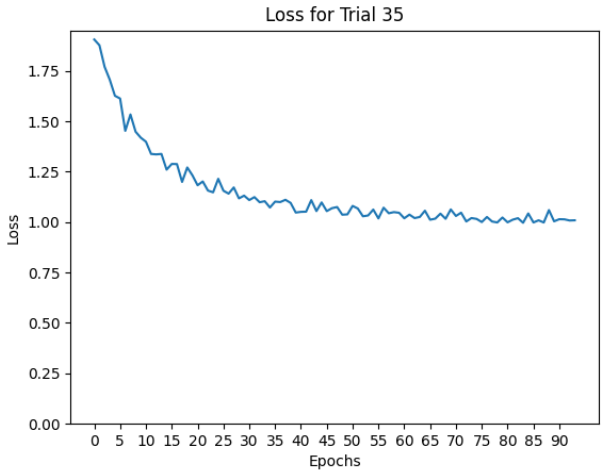

Figure 5.12: Learning curve for the best trial from Scaled-Down Experiment on FER2013.

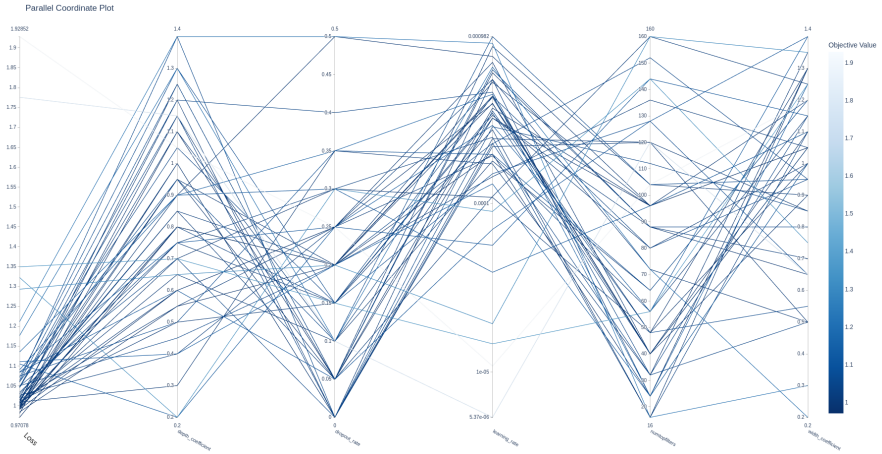

Figure 5.13: Parallel Coordinate plot for Scaled-Down Experiment on FER2013.

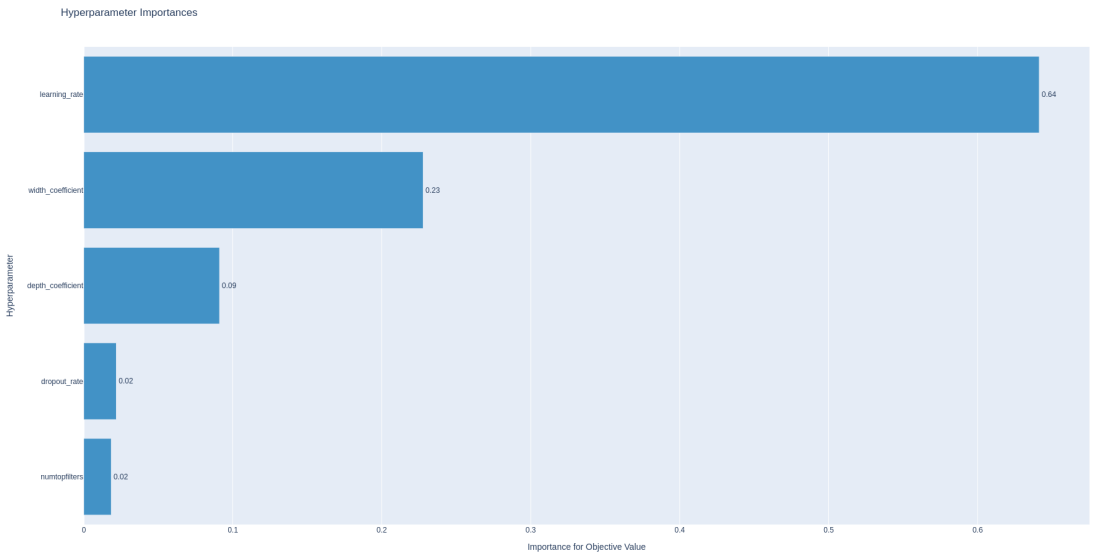

Figure 5.14: Hyperparameter importances for Scaled-Down Experiment on FER2013.

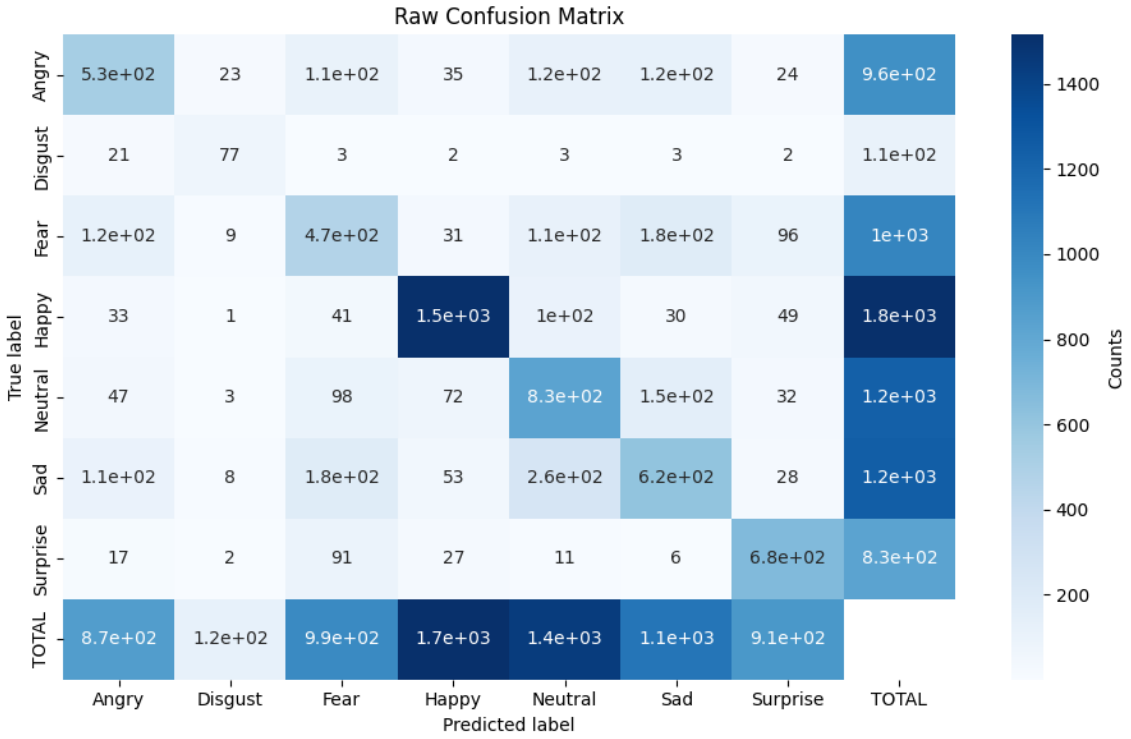

Figure 5.15: Raw Confusion Matrix for Scaled-Down Experiment on FER2013.

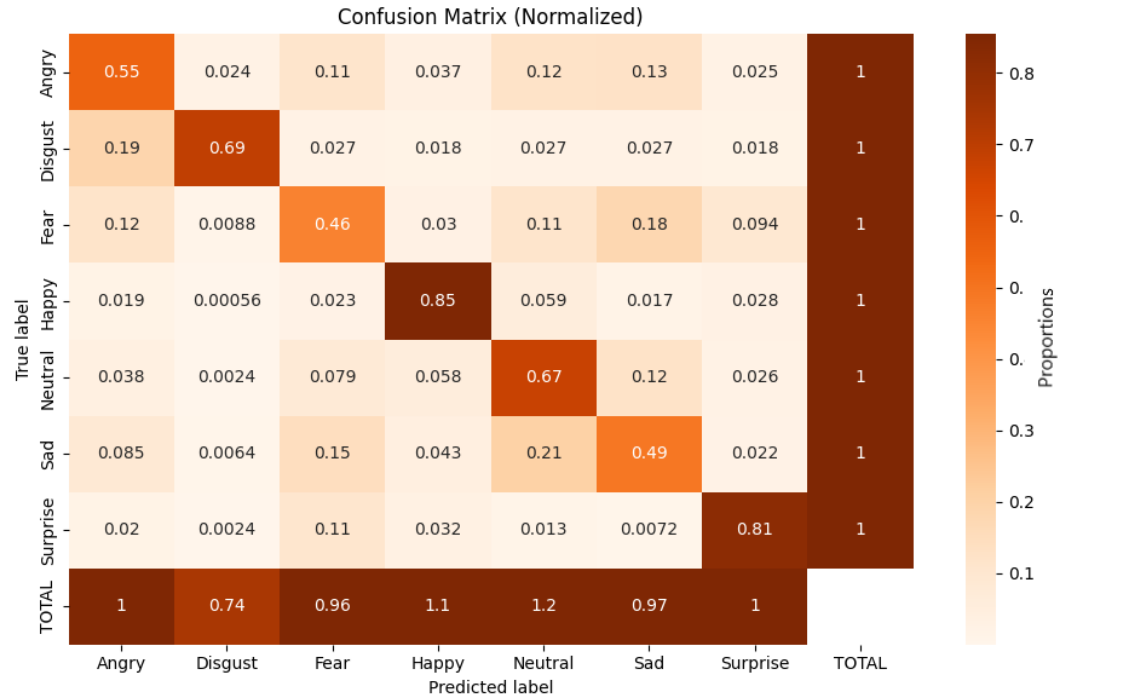

Figure 5.16: Normalized Confusion Matrix for Scaled-Down Experiment on FER2013.

5.4 Scaled-Down Experiment on AffectNet

Table 5.6: Performance Metrics of Scaled-Down Experiment on AffectNet.

Accuracy	0.613
Loss	1.055
Precision	0.613
Recall	0.613
F1Score	0.611

Table 5.7: Best hyperparameter configuration of Scaled-Down Experiment on AffectNet.

Hyperparameter	**Value**
Learning rate	0.0003622
Width coefficient	1.0545
Depth coefficient	1.2
Dropout rate	0.15
numTopFilters	24

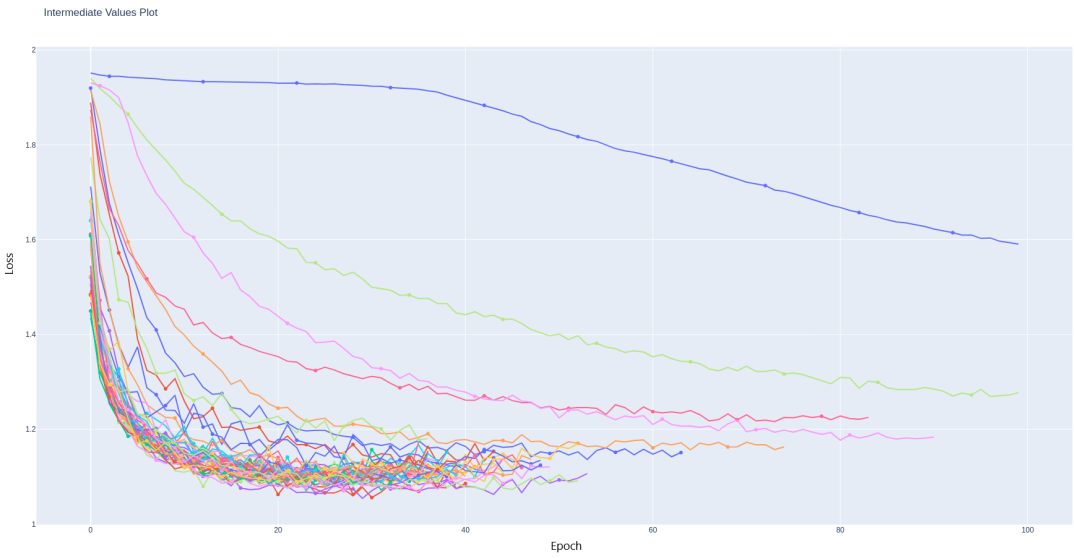

Figure 5.17: Learning curves for every trial from Scaled-Down Experiment on AffectNet.

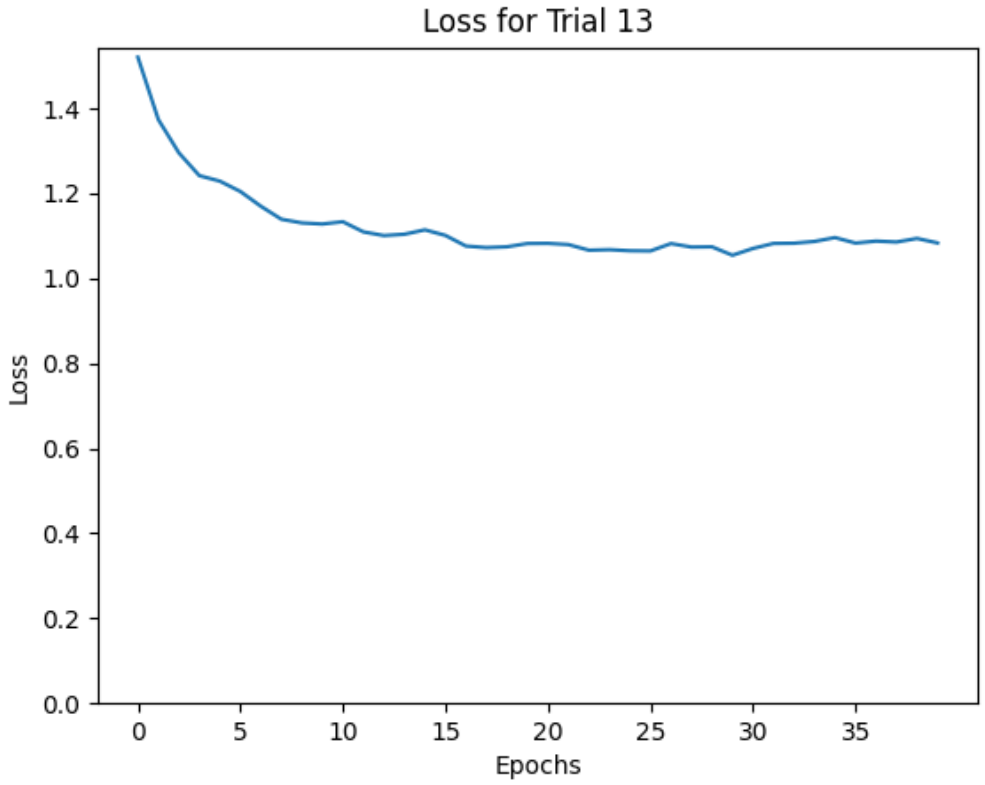

Figure 5.18: Learning curves for the best trial from Scaled-Down Experiment on AffectNet.

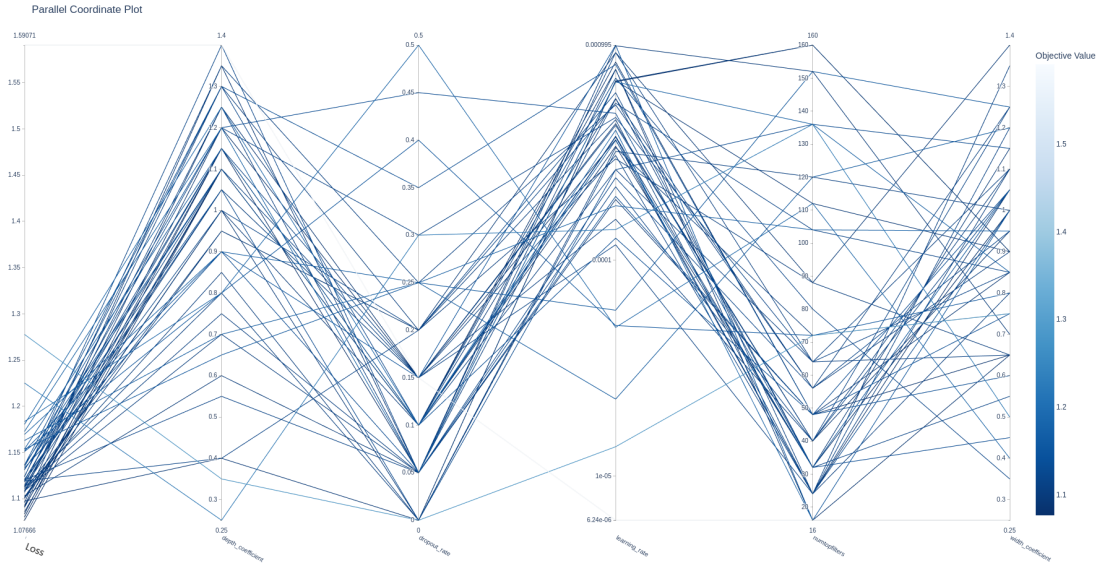

Figure 5.19: Parallel Coordinate plot from Scaled-Down Experiment on AffectNet.

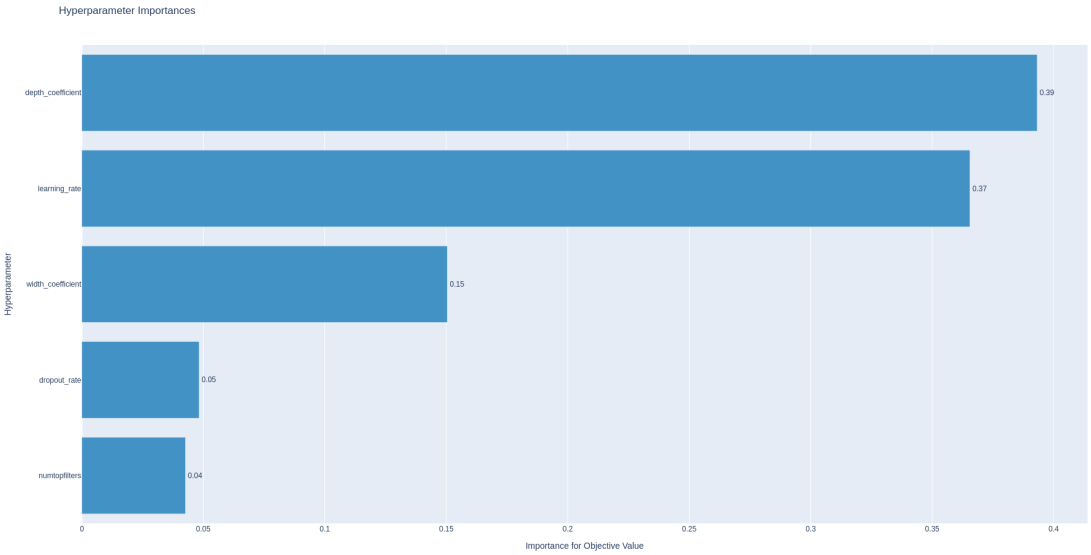

Figure 5.20: Hyperparameter Importances from Scaled-Down Experiment on AffectNet.

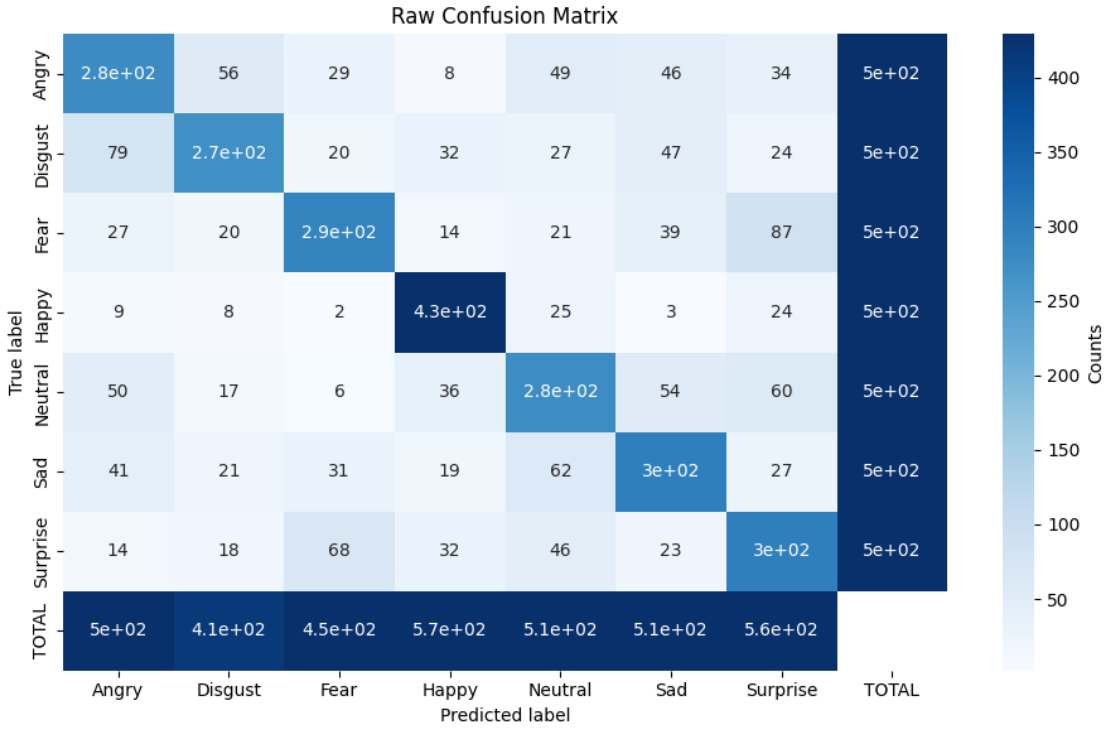

Figure 5.21: Raw Confusion Matrix from Scaled-Down Experiment on AffectNet.

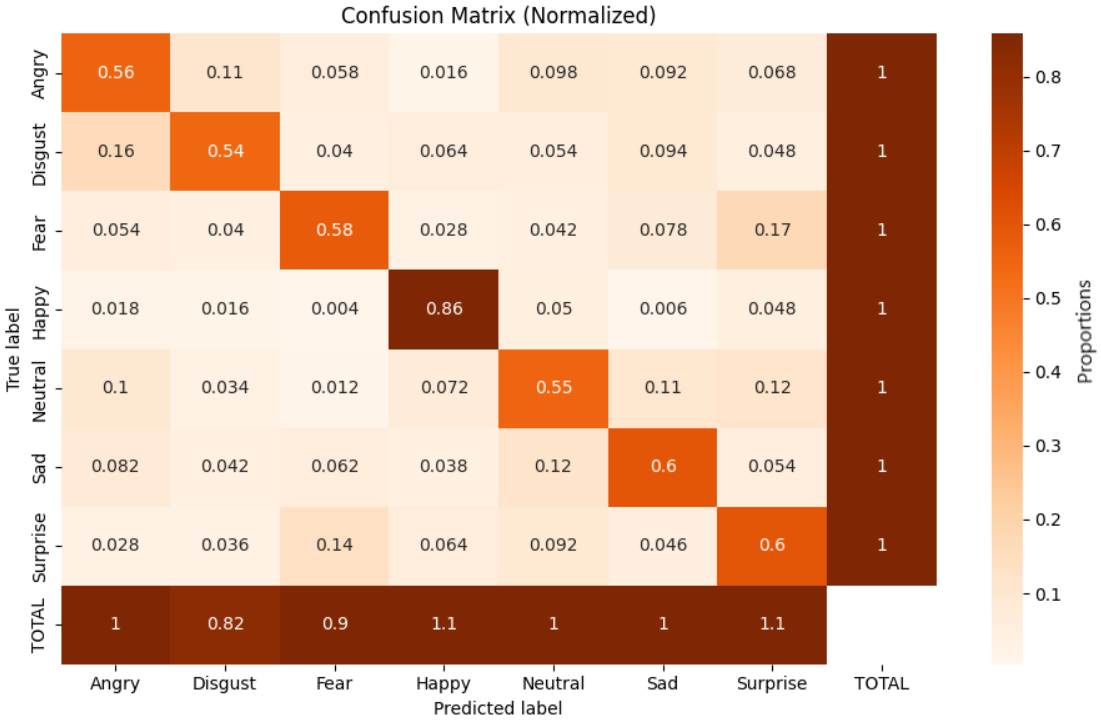

Figure 5.22: Normalized Confusion Matrix from Scaled-Down Experiment on AffectNet.

5.5 Comparison to state-of-the-art

As this study does not involve the use of additional training data or ensembles, a fair comparison will be made by evaluating individual models that, like the current approach, have incorporated class weights without the inclusion of extra training data.

Table 5.8: Comparison to State-of-the-Art Results on FER2013

Model	**Accuracy**
Human-level	65%
ResNet50 WCW [22]	67.7%
SeNET50 WCW [22]	68.9%
VGG16 WCW [22]	70%
Baseline Experiment	65.8%
Scaled-Down Experiment	65.7%

Table 5.9: Comparison to State-of-the-Art Results on AffectNet (7classes).

Model	**Accuracy**
POSTER V2 [25]	67.45%
Scaled-Down Experiment	61.29%

The obtained results demonstrate a notable discrepancy when compared to prior research that excludes 'contempt' and focuses solely on distinguishing among 7 classes on the AffectNet dataset. This discrepancy may be attributed to the reduction in image resolution from the original 224x224 to 48x48 pixels.

Chapter 6

Conclusion

6.1 Achievements

In conclusion, this dissertation made a significant effort to address the recognition and classification of facial expressions using a scaled-down EfficientNetV2S architecture. This approach combined the scaling coefficients of EfficientNetV2S with the use of Optuna, a hyperparameter optimization tool, which allowed for the systematic exploration of both architecture and hyperparameter configurations. One of the primary achievements of this work is the creation of a customized and efficient deep learning architecture tailored to the task of facial expression recognition. The results from the Baseline Experiment on FER2013 demonstrated a human-level performance although marginally trailing behind the state-of-the-art. The Scaled-Down Experiment successfully reduced the number of parameters and training time with negligible impact on performance compared to the Baseline.
However, despite these notable findings, their performance could not match the current state-of-the-art.

6.2 Future Work

With regards to the hypothesis comparing the performance of a specialized FER network to one using transfer learning from ILRSVC, this question remains unresolved. The reason is that designing a specialized deep learning network for FER presents several challenges, including the need for extensive labeled data, computational resources, and domain expertise. Future research should explore ways to mitigate these challenges. Some possible extensions of the current work include the performance of cross-validation with multiple datasets for assessing the generalization performance of FER models. Future research should consider collecting and making use of diverse facial expression datasets to validate model robustness across different demographics, lighting conditions, and cultural contexts. By introducing attention mechanisms into the architecture, the model's capability to emphasize crucial facial regions, a vital component of Facial Expression Recognition (FER), can be enhanced, prompting the need for further exploration of different attention mechanisms and their impact on recognition accuracy in future research. The integration of Generative Adversarial Networks (GANs) for data augmentation is another possible direction for future research. In summary, while this dissertation has achieved notable success in the field of facial expression recognition, there remain numerous avenues for further exploration and improvement.

By addressing the challenges and pursuing the strategies outlined above, future research can continue to advance the state of the art in automatic FER models and contribute to a broader understanding of human emotions.

Bibliography

[1] P. Ekman, "An argument for basic emotions," *Cognition & Emotion*, vol. 6, no. 3-4, pp. 169–200, 1992. 3

[2] J. A. Russell, "A circumplex model of affect." *Journal of Personality and Social Psychology*, vol. 39, no. 6, p. 1161, 1980. 3, 38

[3] A. Mehrabian, "Pleasure-arousal-dominance: A general framework for describing and measuring individual differences in temperament," *Current Psychology*, vol. 14, pp. 261–292, 1996. 4

[4] P. Ramachandran, B. Zoph, and Q. V. Le, "Searching for activation functions," *arXiv preprint arXiv:1710.05941*, 2017. 11

[5] D. Hendrycks and K. Gimpel, "Gaussian error linear units (GELUs)," *arXiv preprint arXiv:1606.08415*, 2016. 11

[6] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014. 14

[7] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: a simple way to prevent neural networks from overfitting," *The Journal of Machine Learning Research*, vol. 15, no. 1, pp. 1929–1958, 2014. 17

[8] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *International Conference on Machine Learning*, 2015, pp. 448–456. 17

[9] S. Santurkar, D. Tsipras, A. Ilyas, and A. Madry, "How does batch normalization help optimization?" *Advances in Neural Information Processing Systems*, vol. 31, 2018. 18

[10] Z. Li, F. Liu, W. Yang, S. Peng, and J. Zhou, "A survey of convolutional neural networks: Analysis, applications, and prospects," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 33, no. 12, pp. 6999–7019, 2022. 18

[11] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, Z. Ma, Se aannd Huang, A. Karpathy, A. Khosla, M. Bernstein *et al.*, "ImageNet large scale visual recognition challenge," *International Journal of Computer Vision*, vol. 115, pp. 211–252, 2015. 25, 33

[12] R. Ayachi, M. Afif, Y. Said, and M. Atri, "Strided convolution instead of max pooling for memory efficiency of convolutional neural networks," in *Proceedings of the 8th International Conference on Sciences of Electronics, Technologies of Information and Telecommunications (SETIT'18), Vol.1*, M. S. Bouhlel and S. Rovetta, Eds. Cham: Springer International Publishing, 2020, pp. 234–243. 28

[13] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 770–778. 28

[14] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, "MobileNetV2: Inverted residuals and linear bottlenecks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 4510–4520. 30

[15] M. Tan and Q. Le, "EfficientNet: Rethinking model scaling for convolutional neural networks," in *International Conference on Machine Learning*, 2019, pp. 6105–6114. 32

[16] ——, "EfficientNetV2: Smaller models and faster training," in *International Conference on Machine Learning*, 2021, pp. 10 096–10 106. 32

[17] B. Zoph, V. Vasudevan, J. Shlens, and Q. V. Le, "Learning transferable architectures for scalable image recognition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 8697–8710. 32

[18] J. Hu, L. Shen, and G. Sun, "Squeeze-and-excitation networks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 7132–7141. 33

[19] S. Li and W. Deng, "Deep facial expression recognition: A survey," *IEEE Transactions on Affective Computing*, vol. 13, no. 3, pp. 1195–1215, 2020. 33

[20] L. Zahara, P. Musa, E. Prasetyo Wibowo, I. Karim, and S. Bahri Musa, "The Facial Emotion Recognition (FER-2013) dataset for prediction system of micro-expressions face using the Convolutional Neural Network (CNN) algorithm based raspberry pi," in *2020 Fifth International Conference on Informatics and Computing (ICIC)*, 2020, pp. 1–9. 36

[21] A. Mollahosseini, B. Hasani, and M. H. Mahoor, "AffectNet: A database for facial expression, valence, and arousal computing in the wild," *IEEE Transactions on Affective Computing*, vol. 10, no. 1, pp. 18–31, 2019. 37

[22] A. Khanzada, C. Bai, and F. T. Celepcikay, "Facial expression recognition with deep learning," *arXiv preprint arXiv:2004.11823*, 2020. 40, 59

[23] T. Akiba, S. Sano, T. Yanase, T. Ohta, and M. Koyama, "Optuna: A next-generation hyperparameter optimization framework," in *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2019, pp. 2623–2631. 42

[24] J. Bergstra, R. Bardenet, Y. Bengio, and B. Kégl, "Algorithms for hyper-parameter optimization," *Advances in Neural Information Processing Systems*, vol. 24, 2011. 43

[25] J. Mao, R. Xu, X. Yin, Y. Chang, B. Nie, and A. Huang, “POSTER V2: A simpler and stronger facial expression recognition network,” *arXiv preprint arXiv:2301.12149*, 2023. 59