NORTHUMBRIA UNIVERSITY

Project Report

Machine Learning-based Automated Facial Emotion Recognition

DECLARATIONS

A report submitted in partial fulfilment of the regulations governing the award of the Degree of BSc. (Honours) Computer Science at the University of Northumbria at Newcastle

Alexandru-Adonis Neagu

2020/2021

Investigative Project

I declare the following:

(1) that the material contained in this dissertation is the end result of my own work and that due acknowledgement has been given in the bibliography and references to ALL sources be they printed, electronic or personal.

(2) the Word Count of this Dissertation is 18612

(3) that unless this dissertation has been confirmed as confidential, I agree to an entire electronic copy or sections of the dissertation to being placed on the eLearning Portal (Blackboard), if deemed appropriate, to allow future students the opportunity to see examples of past dissertations.  I understand that if displayed on eLearning Portal it would be made available for no longer than five years and that students would be able to print off copies or download.

(4) I agree to my dissertation being submitted to a plagiarism detection service, where it will be stored in a database and compared against work submitted from this or any other School or from other institutions using the service.

In the event of the service detecting a high degree of similarity between content within the service this will be reported back to my supervisor and second marker, who may decide to undertake further investigation that may ultimately lead to disciplinary actions, should instances of plagiarism be detected.

(5) I have read the Northumbria University/Engineering and Environment Policy Statement on Ethics in Research and Consultancy and I confirm that ethical issues have been considered, evaluated and appropriately addressed in this research.

SIGNED:

Shape

Description automatically generated

# Acknowledgements



I would like to thank my dissertation supervisor Dr. Ossama Alshabrawy for his support, guidance and particularly his words of encouragement throughout the project.

I would like to thank my Partner, Friends and Family for their support, including to all who participated to my evolution during university



# Abstract

The purpose of this dissertation is to describe the design, implementation, testing, and evaluation of a Facial Expression Recognition (FER) system that utilizes a machine learning algorithm based on Deep Convolutional Neural Networks (DCNNs) to correctly classify seven facial emotions as happiness, sadness, fear, anger, disgust, and neutral. This task was difficult due to the dataset's very imbalanced nature. This results in a bias learning model that is less accurate at predicting minority classes than majority classes.

The DCNN models were developed in Python, using the GPU's processing capabilities to accelerate the training and testing processes. Facial expressions provide useful information that is difficult for a conventional algorithm to detect. Recognizing them, on the other hand, may result in more responsive and smarter systems, which may enhance the user experience.

By testing with several models such as ResNet101V2, MobileNetV2, InceptionV3, and VGG19 and various architectures on the AffectNet dataset, the most appropriate hyper parameters with a high degree of performance were identified. The process of experimenting included finding the right number for convolutional layers, fully connected layers, neurons, learning rates, optimizers and regularization approaches. Additionally, a thorough knowledge of the DCNNs' strengths and weaknesses was obtained.

The results of these studies demonstrate the need of exercising caution at various stages of the development process, such as architectural selection and hyper parameter tweaking. By choosing the optimal mix of these two components, the model's accuracy and convergence time increase. KerasTuner was used to ensure that the right mixture of parameters can be achieved one the architecture was designed.

The optimal model was compared to the top 10 optimal models created by other authors using the AffectNet dataset. The limits of the proscribed job and their commonalities were addressed here. Additionally, it includes a summary of the process of creating the whole project, including goals, impediments, and future ethical, social, and legal considerations.

The paper concludes with a summary of the document's goals and accomplishments, as well as recommendations for future improvements and best practices.

Contents

[Acknowledgements 2](#_Toc80891434)

[Abstract 3](#_Toc80891435)

[List of figures: 7](#_Toc80891436)

[List of tables: 9](#_Toc80891437)

[Introduction 10](#_Toc80891438)

[1.1. Motivation 10](#_Toc80891439)

[1.2. Aims 11](#_Toc80891440)

[1.3. Objectives 11](#_Toc80891441)

[1.3. Structure of the dissertation 11](#_Toc80891442)

[Analysis 12](#_Toc80891443)

[Chapter 2. Related work 12](#_Toc80891444)

[Chapter 3. Literature review 15](#_Toc80891445)

[3.1 General approach to facial emotion recognition 15](#_Toc80891446)

[3.2 Classification 16](#_Toc80891447)

[3.2.1 Description of linear classifiers 16](#_Toc80891448)

[3.2.2 Non-linear classifiers 17](#_Toc80891449)

[3.3 Neural networks 17](#_Toc80891450)

[3.3.1 Feed-forward neural network 17](#_Toc80891451)

[3.4. Activation function 18](#_Toc80891452)

[3.4.1. Tanh 18](#_Toc80891453)

[3.4.2. Sigmoid 18](#_Toc80891454)

[3.4.3. ReLU and Leaky ReLU 19](#_Toc80891455)

[3.4.4.Maxout 19](#_Toc80891456)

[3.5 Gradient descendent optimisation 19](#_Toc80891457)

[3.5.1 Gradient descendent optimisation algorithms 20](#_Toc80891458)

[3.5.2 Backpropagation 22](#_Toc80891459)

[3.4 Challenges of building a model 23](#_Toc80891460)

[3.4.1 Description of Bias-Variance compromise, overfitting and underfitting 23](#_Toc80891461)

[3.4.2 Disappearing gradient and inflating gradient 24](#_Toc80891462)

[3.5 Convolutional Neural Network using Deep Learning 24](#_Toc80891463)

[3.5.1 Convolutional layer 24](#_Toc80891464)

[3.5.2 Pooling or subsampling layer 27](#_Toc80891465)

[3.5.3 Fully connected layer 28](#_Toc80891466)

[3.5.4 Backpropagation in deep convolutional neural network 28](#_Toc80891467)

[3.5.5 Softmax classifier 28](#_Toc80891468)

[3.5.5 Discrete convolution 29](#_Toc80891469)

[3.5.6 Deep convolutional neural network architectures 29](#_Toc80891470)

[3.5.7 Strategies for enhancement 30](#_Toc80891471)

[Synthesis 31](#_Toc80891472)

[Chapter 4. Project research 31](#_Toc80891473)

[4.1 AffectNet dataset 31](#_Toc80891474)

[4.2 Transfer learning 33](#_Toc80891475)

[4.3 Pre-trained modes 33](#_Toc80891476)

[4.3.1 ResNet101V2 model 33](#_Toc80891477)

[4.3.2 VGG19 model 34](#_Toc80891478)

[4.2.3 InceptionV3 model 35](#_Toc80891479)

[4.3.4 MobileNetV2 model 36](#_Toc80891480)

[4.4 Hardware resources 37](#_Toc80891481)

[4.4.1 Personal Computer 37](#_Toc80891482)

[4.4.2 Google Colaboratory web service 37](#_Toc80891483)

[4.5 Programming language 37](#_Toc80891484)

[4.6 Software and libraries 38](#_Toc80891485)

[4.4 Tuning the hyper-parameters 39](#_Toc80891486)

[Chapter 5. Experiments and results 40](#_Toc80891487)

[5.1. First configuration models 40](#_Toc80891488)

[5.2. Second configuration models 44](#_Toc80891489)

[5.3. Third configuration models 46](#_Toc80891490)

[5.4. Fourth configuration models 50](#_Toc80891491)

[5.5. Fifth configuration models 54](#_Toc80891492)

[5.6. Sixth configuration models 57](#_Toc80891493)

[5.7. Seventh configuration models 59](#_Toc80891494)

[5.8. Eighth configuration models 62](#_Toc80891495)

[Evaluation and conclusion 65](#_Toc80891496)

[Chapter 6. Evaluation 65](#_Toc80891497)

[6.2. Project evaluation 68](#_Toc80891498)

[Chapter 7. Conclusion and recommendation 69](#_Toc80891499)

[References 70](#_Toc80891500)

[Appendix A: Terms of reference (TOR) 76](#_Toc80891501)

[Appendix B: Project Initiation Document (PID) 84](#_Toc80891502)

[Appendix C: Import all the libraries required and configure the GPU for training 85](#_Toc80891503)

[Appendix D: Import a pre-trained model from Keras using the weights from ImageNet 86](#_Toc80891504)

[Appendix D: Use ImageDataGenerator() for data augmentation 87](#_Toc80891505)

[Appendix E: First configuration stage 87](#_Toc80891506)

[Appendix F: Second configuration stage 87](#_Toc80891507)

[Appendix G: Third configuration stage-Adagrad 88](#_Toc80891508)

[Appendix H: Third configuration stage-Adadelta 88](#_Toc80891509)

[Appendix I: Fourth configuration stage-learning rate 0.5 88](#_Toc80891510)

[Appendix J: Fourth configuration stage-learning rate 0.01 89](#_Toc80891511)

[Appendix J: Fifth configuration stage 89](#_Toc80891512)

[Appendix K: Sixth configuration stage 90](#_Toc80891513)

[Appendix L: Seventh configuration stage 90](#_Toc80891514)

[Appendix M: Eighth configuration stage 91](#_Toc80891515)

[Appendix N: initialize the tuner to find the best model 92](#_Toc80891516)

[Appendix O: initialize the tuner to find the best model 92](#_Toc80891517)

[Appendix P: Train the model and save the best weights using ModelCheckpoint 92](#_Toc80891518)

[Appendix Q: Plot the accuracy and cross-entropy diagrams 92](#_Toc80891519)

[Appendix R: Import all the libraries required and configure the GPU for training 93](#_Toc80891520)

[Appendix S: Use ImageDataGenerator() for data augmentation in testing 93](#_Toc80891521)

[Appendix T: Create a list with the label of each picture 94](#_Toc80891522)

[Appendix U: Create a result map 94](#_Toc80891523)

[Appendix V: Make prediction on the validation set and create a list with the name of the emotion predicted 95](#_Toc80891524)

[Appendix W: Create and plot confusion matrix 95](#_Toc80891525)

[Appendix X: Create and print the classification report 95](#_Toc80891526)

[Appendix Y: Create and plot the ROC curve for each class 95](#_Toc80891527)

[Appendix Z: Create and plot the PR curve for each class 96](#_Toc80891528)

# List of figures:

[Figure 1.Facial emotion recognition process 15](#_Toc80666890)

[Figure 2.Bias-variance compromise 23](https://livenorthumbriaac-my.sharepoint.com/personal/w18017575_northumbria_ac_uk/Documents/Desktop/Dissertation2.0.docx#_Toc80666891)

[Figure 3.Local connectivity and receptive field 25](https://livenorthumbriaac-my.sharepoint.com/personal/w18017575_northumbria_ac_uk/Documents/Desktop/Dissertation2.0.docx#_Toc80666892)

[Figure 4.Parameter sharing 25](https://livenorthumbriaac-my.sharepoint.com/personal/w18017575_northumbria_ac_uk/Documents/Desktop/Dissertation2.0.docx#_Toc80666893)

[Figure 5.Activation maps 26](https://livenorthumbriaac-my.sharepoint.com/personal/w18017575_northumbria_ac_uk/Documents/Desktop/Dissertation2.0.docx#_Toc80666894)

[Figure 6.Spatial size reduction 27](https://livenorthumbriaac-my.sharepoint.com/personal/w18017575_northumbria_ac_uk/Documents/Desktop/Dissertation2.0.docx#_Toc80666895)

[Figure 7.Max pooling 27](https://livenorthumbriaac-my.sharepoint.com/personal/w18017575_northumbria_ac_uk/Documents/Desktop/Dissertation2.0.docx#_Toc80666896)

[Figure 8.Down sampling 27](https://livenorthumbriaac-my.sharepoint.com/personal/w18017575_northumbria_ac_uk/Documents/Desktop/Dissertation2.0.docx#_Toc80666897)

[Figure 9.Example of discrete convolution. 29](https://livenorthumbriaac-my.sharepoint.com/personal/w18017575_northumbria_ac_uk/Documents/Desktop/Dissertation2.0.docx#_Toc80666898)

[Figure 10.Data augmentation procedure 31](https://livenorthumbriaac-my.sharepoint.com/personal/w18017575_northumbria_ac_uk/Documents/Desktop/Dissertation2.0.docx#_Toc80666899)

[Figure 11.Representation of an Inception block 35](https://livenorthumbriaac-my.sharepoint.com/personal/w18017575_northumbria_ac_uk/Documents/Desktop/Dissertation2.0.docx#_Toc80666900)

[Figure 12.Representation of depthwise and pointwise convolutions 36](https://livenorthumbriaac-my.sharepoint.com/personal/w18017575_northumbria_ac_uk/Documents/Desktop/Dissertation2.0.docx#_Toc80666901)

[Figure 13.Training results for each model in first stage 41](#_Toc80666902)

[Figure 14.InceptionV3-first stage 42](https://livenorthumbriaac-my.sharepoint.com/personal/w18017575_northumbria_ac_uk/Documents/Desktop/Dissertation2.0.docx#_Toc80666903)

[Figure 15.MobileNetV2-first stage 42](https://livenorthumbriaac-my.sharepoint.com/personal/w18017575_northumbria_ac_uk/Documents/Desktop/Dissertation2.0.docx#_Toc80666904)

[Figure 16.VGG19-first stage 41](https://livenorthumbriaac-my.sharepoint.com/personal/w18017575_northumbria_ac_uk/Documents/Desktop/Dissertation2.0.docx#_Toc80666905)

[Figure 17.ResNet101V2-first stage 41](https://livenorthumbriaac-my.sharepoint.com/personal/w18017575_northumbria_ac_uk/Documents/Desktop/Dissertation2.0.docx#_Toc80666906)

[Figure 18.ResNet101V2-Confusion matrix-first stage 42](https://livenorthumbriaac-my.sharepoint.com/personal/w18017575_northumbria_ac_uk/Documents/Desktop/Dissertation2.0.docx#_Toc80666907)

[Figure 19.VGG19-Confusion matrix-first stage 42](https://livenorthumbriaac-my.sharepoint.com/personal/w18017575_northumbria_ac_uk/Documents/Desktop/Dissertation2.0.docx#_Toc80666908)

[Figure 20.MobileNetV2-Confusion matrix-first stage 42](https://livenorthumbriaac-my.sharepoint.com/personal/w18017575_northumbria_ac_uk/Documents/Desktop/Dissertation2.0.docx#_Toc80666909)

[Figure 21.InceptionV3-Confusion matrix-first stage 42](https://livenorthumbriaac-my.sharepoint.com/personal/w18017575_northumbria_ac_uk/Documents/Desktop/Dissertation2.0.docx#_Toc80666910)

[Figure 22.InceptionV3-Clasification report-first stage 43](https://livenorthumbriaac-my.sharepoint.com/personal/w18017575_northumbria_ac_uk/Documents/Desktop/Dissertation2.0.docx#_Toc80666911)

[Figure 23.MobileNetV2-Clasification report-first stage 43](https://livenorthumbriaac-my.sharepoint.com/personal/w18017575_northumbria_ac_uk/Documents/Desktop/Dissertation2.0.docx#_Toc80666912)

[Figure 24.ResNet101V2-Clasification report-first stage 43](https://livenorthumbriaac-my.sharepoint.com/personal/w18017575_northumbria_ac_uk/Documents/Desktop/Dissertation2.0.docx#_Toc80666913)

[Figure 25.VGG19-Clasification report-first stage 43](https://livenorthumbriaac-my.sharepoint.com/personal/w18017575_northumbria_ac_uk/Documents/Desktop/Dissertation2.0.docx#_Toc80666914)

[Figure 26.Training results for each model in second stage 44](https://livenorthumbriaac-my.sharepoint.com/personal/w18017575_northumbria_ac_uk/Documents/Desktop/Dissertation2.0.docx#_Toc80666915)

[Figure 27.InceptionV3-seond stage 44](https://livenorthumbriaac-my.sharepoint.com/personal/w18017575_northumbria_ac_uk/Documents/Desktop/Dissertation2.0.docx#_Toc80666916)

[Figure 28.MobileNetV2-second stage 44](https://livenorthumbriaac-my.sharepoint.com/personal/w18017575_northumbria_ac_uk/Documents/Desktop/Dissertation2.0.docx#_Toc80666917)

[Figure 29.VGG19-second stage 44](https://livenorthumbriaac-my.sharepoint.com/personal/w18017575_northumbria_ac_uk/Documents/Desktop/Dissertation2.0.docx#_Toc80666918)

[Figure 30.ResNet101V2-second stage 44](https://livenorthumbriaac-my.sharepoint.com/personal/w18017575_northumbria_ac_uk/Documents/Desktop/Dissertation2.0.docx#_Toc80666919)

[Figure 31.InceptionV3-Confusion matrix-second stage 45](https://livenorthumbriaac-my.sharepoint.com/personal/w18017575_northumbria_ac_uk/Documents/Desktop/Dissertation2.0.docx#_Toc80666920)

[Figure 32.MobileNetV2-Confusion matrix-second stage 45](https://livenorthumbriaac-my.sharepoint.com/personal/w18017575_northumbria_ac_uk/Documents/Desktop/Dissertation2.0.docx#_Toc80666921)

[Figure 33.ResNet101V2-Confusion matrix-second stage 45](https://livenorthumbriaac-my.sharepoint.com/personal/w18017575_northumbria_ac_uk/Documents/Desktop/Dissertation2.0.docx#_Toc80666922)

[Figure 34.VGG19-Confusion matrix-second stage 45](https://livenorthumbriaac-my.sharepoint.com/personal/w18017575_northumbria_ac_uk/Documents/Desktop/Dissertation2.0.docx#_Toc80666923)

[Figure 35.ResNet101V2-Clasification report-second stage 45](https://livenorthumbriaac-my.sharepoint.com/personal/w18017575_northumbria_ac_uk/Documents/Desktop/Dissertation2.0.docx#_Toc80666924)

[Figure 36.VGG19-Clasification report-second stage 45](https://livenorthumbriaac-my.sharepoint.com/personal/w18017575_northumbria_ac_uk/Documents/Desktop/Dissertation2.0.docx#_Toc80666925)

[Figure 37.Inceptionv3-Clasification report-second stage 46](https://livenorthumbriaac-my.sharepoint.com/personal/w18017575_northumbria_ac_uk/Documents/Desktop/Dissertation2.0.docx#_Toc80666926)

[Figure 38.MobileNetV2-Claification report-second stage 46](https://livenorthumbriaac-my.sharepoint.com/personal/w18017575_northumbria_ac_uk/Documents/Desktop/Dissertation2.0.docx#_Toc80666927)

[Figure 39.Optimizer comparison for each model 46](https://livenorthumbriaac-my.sharepoint.com/personal/w18017575_northumbria_ac_uk/Documents/Desktop/Dissertation2.0.docx#_Toc80666928)

[Figure 40.Optimization comparison 46](https://livenorthumbriaac-my.sharepoint.com/personal/w18017575_northumbria_ac_uk/Documents/Desktop/Dissertation2.0.docx#_Toc80666929)

[Figure 41.ResNet101V2-Adagrad 46](https://livenorthumbriaac-my.sharepoint.com/personal/w18017575_northumbria_ac_uk/Documents/Desktop/Dissertation2.0.docx#_Toc80666930)

[Figure 42.VGG19-Adagrad 46](https://livenorthumbriaac-my.sharepoint.com/personal/w18017575_northumbria_ac_uk/Documents/Desktop/Dissertation2.0.docx#_Toc80666931)

[Figure 43.InceptionV3-Confusion matrix-Adagrad 47](https://livenorthumbriaac-my.sharepoint.com/personal/w18017575_northumbria_ac_uk/Documents/Desktop/Dissertation2.0.docx#_Toc80666932)

[Figure 44.MobileNetV2-Confusion matrix-Adagrad 47](https://livenorthumbriaac-my.sharepoint.com/personal/w18017575_northumbria_ac_uk/Documents/Desktop/Dissertation2.0.docx#_Toc80666933)

[Figure 45.ResNet101V2-Confusion matrix-Adagrad 47](https://livenorthumbriaac-my.sharepoint.com/personal/w18017575_northumbria_ac_uk/Documents/Desktop/Dissertation2.0.docx#_Toc80666934)

[Figure 46.VGG19-Confusion matrix-Adagrad 47](https://livenorthumbriaac-my.sharepoint.com/personal/w18017575_northumbria_ac_uk/Documents/Desktop/Dissertation2.0.docx#_Toc80666935)

[Figure 47.InceptionV3-Adagrad 47](https://livenorthumbriaac-my.sharepoint.com/personal/w18017575_northumbria_ac_uk/Documents/Desktop/Dissertation2.0.docx#_Toc80666936)

[Figure 48.MobileNetV2-Adagrad 47](https://livenorthumbriaac-my.sharepoint.com/personal/w18017575_northumbria_ac_uk/Documents/Desktop/Dissertation2.0.docx#_Toc80666937)

[Figure 49.InceptionV3-Adadelta 48](https://livenorthumbriaac-my.sharepoint.com/personal/w18017575_northumbria_ac_uk/Documents/Desktop/Dissertation2.0.docx#_Toc80666938)

[Figure 50.MobileNetV2-Adadelta 48](https://livenorthumbriaac-my.sharepoint.com/personal/w18017575_northumbria_ac_uk/Documents/Desktop/Dissertation2.0.docx#_Toc80666939)

[Figure 51.RestNet101V2-Adadelta 48](https://livenorthumbriaac-my.sharepoint.com/personal/w18017575_northumbria_ac_uk/Documents/Desktop/Dissertation2.0.docx#_Toc80666940)

[Figure 52.VGG19-Adadelta 48](https://livenorthumbriaac-my.sharepoint.com/personal/w18017575_northumbria_ac_uk/Documents/Desktop/Dissertation2.0.docx#_Toc80666941)

[Figure 53.InceptionV3-Clasificaion report-Adagrad 48](https://livenorthumbriaac-my.sharepoint.com/personal/w18017575_northumbria_ac_uk/Documents/Desktop/Dissertation2.0.docx#_Toc80666942)

[Figure 54.MobileNet2-Clasification report-Adagrad 48](https://livenorthumbriaac-my.sharepoint.com/personal/w18017575_northumbria_ac_uk/Documents/Desktop/Dissertation2.0.docx#_Toc80666943)

[Figure 55.ResNet101V2-Clasification report-Adagrad 47](https://livenorthumbriaac-my.sharepoint.com/personal/w18017575_northumbria_ac_uk/Documents/Desktop/Dissertation2.0.docx#_Toc80666944)

[Figure 56.VGG19-Clasification report-Adagrad 47](https://livenorthumbriaac-my.sharepoint.com/personal/w18017575_northumbria_ac_uk/Documents/Desktop/Dissertation2.0.docx#_Toc80666945)

[Figure 57.ResNet101V2-Confusion matrix-Adadelta 49](https://livenorthumbriaac-my.sharepoint.com/personal/w18017575_northumbria_ac_uk/Documents/Desktop/Dissertation2.0.docx#_Toc80666946)

[Figure 58.InceptionV2-Confusion matrix-Adadelta 49](https://livenorthumbriaac-my.sharepoint.com/personal/w18017575_northumbria_ac_uk/Documents/Desktop/Dissertation2.0.docx#_Toc80666947)

[Figure 59.MobileNetV2-Clasification report-Adadelta 49](https://livenorthumbriaac-my.sharepoint.com/personal/w18017575_northumbria_ac_uk/Documents/Desktop/Dissertation2.0.docx#_Toc80666948)

[Figure 60.InceptionV3-Clasification report-Adadelta 49](https://livenorthumbriaac-my.sharepoint.com/personal/w18017575_northumbria_ac_uk/Documents/Desktop/Dissertation2.0.docx#_Toc80666949)

[Figure 61.ResNet101V2-Clasification report-Adadelta 49](https://livenorthumbriaac-my.sharepoint.com/personal/w18017575_northumbria_ac_uk/Documents/Desktop/Dissertation2.0.docx#_Toc80666950)

[Figure 62.VGG19-Clasification report-Adadelta 49](https://livenorthumbriaac-my.sharepoint.com/personal/w18017575_northumbria_ac_uk/Documents/Desktop/Dissertation2.0.docx#_Toc80666951)

[Figure 63.MobileNetV2-Confusion matrix-Adadelta 49](https://livenorthumbriaac-my.sharepoint.com/personal/w18017575_northumbria_ac_uk/Documents/Desktop/Dissertation2.0.docx#_Toc80666952)

[Figure 64.VGG19-Confusion matrix-Adadelta 49](https://livenorthumbriaac-my.sharepoint.com/personal/w18017575_northumbria_ac_uk/Documents/Desktop/Dissertation2.0.docx#_Toc80666953)

[Figure 65.Learning rate comparison for each model 50](https://livenorthumbriaac-my.sharepoint.com/personal/w18017575_northumbria_ac_uk/Documents/Desktop/Dissertation2.0.docx#_Toc80666954)

[Figure 66.InceptionV3-learnng rate 0.5 51](https://livenorthumbriaac-my.sharepoint.com/personal/w18017575_northumbria_ac_uk/Documents/Desktop/Dissertation2.0.docx#_Toc80666955)

[Figure 67.MobileNetV2-learning rate 0.5 51](https://livenorthumbriaac-my.sharepoint.com/personal/w18017575_northumbria_ac_uk/Documents/Desktop/Dissertation2.0.docx#_Toc80666956)

[Figure 68.ResNet101V2-learning rate 0.5 51](https://livenorthumbriaac-my.sharepoint.com/personal/w18017575_northumbria_ac_uk/Documents/Desktop/Dissertation2.0.docx#_Toc80666957)

[Figure 69.VGG19-lerarnig rate 0.5 51](https://livenorthumbriaac-my.sharepoint.com/personal/w18017575_northumbria_ac_uk/Documents/Desktop/Dissertation2.0.docx#_Toc80666958)

[Figure 70.ResNet101V2-Confusion matrix-learning rate 0.5 51](https://livenorthumbriaac-my.sharepoint.com/personal/w18017575_northumbria_ac_uk/Documents/Desktop/Dissertation2.0.docx#_Toc80666959)

[Figure 71.InceptionV3-Confusion matrix-learning rate 0.5 51](https://livenorthumbriaac-my.sharepoint.com/personal/w18017575_northumbria_ac_uk/Documents/Desktop/Dissertation2.0.docx#_Toc80666960)

[Figure 72.MobileNetV2-Confusion matrix-learning rate 0.5 51](https://livenorthumbriaac-my.sharepoint.com/personal/w18017575_northumbria_ac_uk/Documents/Desktop/Dissertation2.0.docx#_Toc80666961)

[Figure 73.VGG19-Confusion matrix-learning rate 0.5 51](https://livenorthumbriaac-my.sharepoint.com/personal/w18017575_northumbria_ac_uk/Documents/Desktop/Dissertation2.0.docx#_Toc80666962)

[Figure 74.InceptionV3-Clasification report-learnigrate 0.5 52](https://livenorthumbriaac-my.sharepoint.com/personal/w18017575_northumbria_ac_uk/Documents/Desktop/Dissertation2.0.docx#_Toc80666963)

[Figure 75.MobileNetV2-Clasification report-learnig rate 0.5 52](https://livenorthumbriaac-my.sharepoint.com/personal/w18017575_northumbria_ac_uk/Documents/Desktop/Dissertation2.0.docx#_Toc80666964)

[Figure 76.ResNet101V2-Clasification report-learning rate 0.5 52](https://livenorthumbriaac-my.sharepoint.com/personal/w18017575_northumbria_ac_uk/Documents/Desktop/Dissertation2.0.docx#_Toc80666965)

[Figure 77.VGG19-Clasification report-learning rate 0.5 52](https://livenorthumbriaac-my.sharepoint.com/personal/w18017575_northumbria_ac_uk/Documents/Desktop/Dissertation2.0.docx#_Toc80666966)

[Figure 78.ResNet101V2-learning rate 0.01 52](https://livenorthumbriaac-my.sharepoint.com/personal/w18017575_northumbria_ac_uk/Documents/Desktop/Dissertation2.0.docx#_Toc80666967)

[Figure 79.VGG19-learning rate 0.01 52](https://livenorthumbriaac-my.sharepoint.com/personal/w18017575_northumbria_ac_uk/Documents/Desktop/Dissertation2.0.docx#_Toc80666968)

[Figure 80.MobileNetV2-learning rate 0.01 53](https://livenorthumbriaac-my.sharepoint.com/personal/w18017575_northumbria_ac_uk/Documents/Desktop/Dissertation2.0.docx#_Toc80666969)

[Figure 81.InceptionV3-learning rate 0.01 53](https://livenorthumbriaac-my.sharepoint.com/personal/w18017575_northumbria_ac_uk/Documents/Desktop/Dissertation2.0.docx#_Toc80666970)

[Figure 82.InceptionV3-Confusion matrix-learning rate 0.01 53](https://livenorthumbriaac-my.sharepoint.com/personal/w18017575_northumbria_ac_uk/Documents/Desktop/Dissertation2.0.docx#_Toc80666971)

[Figure 83.ResNet101V2-Confusion matrix-learning rate 0.01 53](https://livenorthumbriaac-my.sharepoint.com/personal/w18017575_northumbria_ac_uk/Documents/Desktop/Dissertation2.0.docx#_Toc80666972)

[Figure 84.VGG19-Confusion matrix-learning rate 0.01 53](https://livenorthumbriaac-my.sharepoint.com/personal/w18017575_northumbria_ac_uk/Documents/Desktop/Dissertation2.0.docx#_Toc80666973)

[Figure 85.MobileNetV2-Confusion matrix-0.01 53](https://livenorthumbriaac-my.sharepoint.com/personal/w18017575_northumbria_ac_uk/Documents/Desktop/Dissertation2.0.docx#_Toc80666974)

[Figure 86.InceptionV3-Clasification report-learning rate 0.01 54](https://livenorthumbriaac-my.sharepoint.com/personal/w18017575_northumbria_ac_uk/Documents/Desktop/Dissertation2.0.docx#_Toc80666975)

[Figure 87.VGG19-Clasification report-learning rate 0.01 54](https://livenorthumbriaac-my.sharepoint.com/personal/w18017575_northumbria_ac_uk/Documents/Desktop/Dissertation2.0.docx#_Toc80666976)

[Figure 88.MobileNetV2-Clasification report-learning rate 0.01 54](https://livenorthumbriaac-my.sharepoint.com/personal/w18017575_northumbria_ac_uk/Documents/Desktop/Dissertation2.0.docx#_Toc80666977)

[Figure 89.ResNet101V2-Clasification report-learning rate 0.01 54](https://livenorthumbriaac-my.sharepoint.com/personal/w18017575_northumbria_ac_uk/Documents/Desktop/Dissertation2.0.docx#_Toc80666978)

[Figure 90.Training results for each model in the fifth stage 55](#_Toc80666979)

[Figure 91.InceptionV3-fifth stage 55](https://livenorthumbriaac-my.sharepoint.com/personal/w18017575_northumbria_ac_uk/Documents/Desktop/Dissertation2.0.docx#_Toc80666980)

[Figure 92.MobileNetV2-fifth stage 55](https://livenorthumbriaac-my.sharepoint.com/personal/w18017575_northumbria_ac_uk/Documents/Desktop/Dissertation2.0.docx#_Toc80666981)

[Figure 93.ResNet101V2-fifth stage 55](https://livenorthumbriaac-my.sharepoint.com/personal/w18017575_northumbria_ac_uk/Documents/Desktop/Dissertation2.0.docx#_Toc80666982)

[Figure 94.VGG19-fifth stage 55](https://livenorthumbriaac-my.sharepoint.com/personal/w18017575_northumbria_ac_uk/Documents/Desktop/Dissertation2.0.docx#_Toc80666983)

[Figure 95.InceptionV3-Clsification report-fifth stage 56](https://livenorthumbriaac-my.sharepoint.com/personal/w18017575_northumbria_ac_uk/Documents/Desktop/Dissertation2.0.docx#_Toc80666984)

[Figure 96.MobileNetV2-Clasification report-fifth stage 56](https://livenorthumbriaac-my.sharepoint.com/personal/w18017575_northumbria_ac_uk/Documents/Desktop/Dissertation2.0.docx#_Toc80666985)

[Figure 97.ResNet101V2-Clasification report-fifth stage 56](https://livenorthumbriaac-my.sharepoint.com/personal/w18017575_northumbria_ac_uk/Documents/Desktop/Dissertation2.0.docx#_Toc80666986)

[Figure 98.VGG19-Clasification report-fifth stage 56](https://livenorthumbriaac-my.sharepoint.com/personal/w18017575_northumbria_ac_uk/Documents/Desktop/Dissertation2.0.docx#_Toc80666987)

[Figure 99.InceptionV3-Confusion matrix-fifth stage 56](https://livenorthumbriaac-my.sharepoint.com/personal/w18017575_northumbria_ac_uk/Documents/Desktop/Dissertation2.0.docx#_Toc80666988)

[Figure 100.MobileNetV2-Confusion matrix-fifth stage 56](https://livenorthumbriaac-my.sharepoint.com/personal/w18017575_northumbria_ac_uk/Documents/Desktop/Dissertation2.0.docx#_Toc80666989)

[Figure 101.ResNet101V2-Confusion matrix-fifth stage 56](https://livenorthumbriaac-my.sharepoint.com/personal/w18017575_northumbria_ac_uk/Documents/Desktop/Dissertation2.0.docx#_Toc80666990)

[Figure 102.VGG19-Confusion matrix-fifth stage 56](https://livenorthumbriaac-my.sharepoint.com/personal/w18017575_northumbria_ac_uk/Documents/Desktop/Dissertation2.0.docx#_Toc80666991)

[Figure 103.Training results for each model in the sixth stage 57](https://livenorthumbriaac-my.sharepoint.com/personal/w18017575_northumbria_ac_uk/Documents/Desktop/Dissertation2.0.docx#_Toc80666992)

[Figure 104.ResNet101V2-sixth stage 57](https://livenorthumbriaac-my.sharepoint.com/personal/w18017575_northumbria_ac_uk/Documents/Desktop/Dissertation2.0.docx#_Toc80666993)

[Figure 105.VGG19-sixth stage 57](https://livenorthumbriaac-my.sharepoint.com/personal/w18017575_northumbria_ac_uk/Documents/Desktop/Dissertation2.0.docx#_Toc80666994)

[Figure 106.InceptionV3-sixth stage 58](https://livenorthumbriaac-my.sharepoint.com/personal/w18017575_northumbria_ac_uk/Documents/Desktop/Dissertation2.0.docx#_Toc80666995)

[Figure 107.MobileNetV2-sixth stage 58](https://livenorthumbriaac-my.sharepoint.com/personal/w18017575_northumbria_ac_uk/Documents/Desktop/Dissertation2.0.docx#_Toc80666996)

[Figure 108.ResNetV2-Confusion matrix-sixth stage 58](https://livenorthumbriaac-my.sharepoint.com/personal/w18017575_northumbria_ac_uk/Documents/Desktop/Dissertation2.0.docx#_Toc80666997)

[Figure 109.VGG19-Confusion matrix-sixth stage 58](https://livenorthumbriaac-my.sharepoint.com/personal/w18017575_northumbria_ac_uk/Documents/Desktop/Dissertation2.0.docx#_Toc80666998)

[Figure 110.IncptionV3-Confusion matrix-sixth stage 58](https://livenorthumbriaac-my.sharepoint.com/personal/w18017575_northumbria_ac_uk/Documents/Desktop/Dissertation2.0.docx#_Toc80666999)

[Figure 111.MobileNetV2-Confusion matrix-sixth stage 58](https://livenorthumbriaac-my.sharepoint.com/personal/w18017575_northumbria_ac_uk/Documents/Desktop/Dissertation2.0.docx#_Toc80667000)

[Figure 112.Inceptionv3-Clasification report-sixth stage 59](https://livenorthumbriaac-my.sharepoint.com/personal/w18017575_northumbria_ac_uk/Documents/Desktop/Dissertation2.0.docx#_Toc80667001)

[Figure 113.MobileNetV2-Clasification report-sixth stage 59](https://livenorthumbriaac-my.sharepoint.com/personal/w18017575_northumbria_ac_uk/Documents/Desktop/Dissertation2.0.docx#_Toc80667002)

[Figure 114.ResNet101V2-Clasification report-sixth stage 59](https://livenorthumbriaac-my.sharepoint.com/personal/w18017575_northumbria_ac_uk/Documents/Desktop/Dissertation2.0.docx#_Toc80667003)

[Figure 115.VGG19-Clasification report-sixth stage 59](https://livenorthumbriaac-my.sharepoint.com/personal/w18017575_northumbria_ac_uk/Documents/Desktop/Dissertation2.0.docx#_Toc80667004)

[Figure 116.Training results for each model in the seventh stage 60](https://livenorthumbriaac-my.sharepoint.com/personal/w18017575_northumbria_ac_uk/Documents/Desktop/Dissertation2.0.docx#_Toc80667005)

[Figure 117.ResNet101V2-seventh stage 60](https://livenorthumbriaac-my.sharepoint.com/personal/w18017575_northumbria_ac_uk/Documents/Desktop/Dissertation2.0.docx#_Toc80667006)

[Figure 118.VGG19-seventh stage 60](https://livenorthumbriaac-my.sharepoint.com/personal/w18017575_northumbria_ac_uk/Documents/Desktop/Dissertation2.0.docx#_Toc80667007)

[Figure 119.InceptionV3-seventh stage 61](https://livenorthumbriaac-my.sharepoint.com/personal/w18017575_northumbria_ac_uk/Documents/Desktop/Dissertation2.0.docx#_Toc80667008)

[Figure 120.MobileNetV2-seventh stage 61](https://livenorthumbriaac-my.sharepoint.com/personal/w18017575_northumbria_ac_uk/Documents/Desktop/Dissertation2.0.docx#_Toc80667009)

[Figure 121.InceptionV3-Confusion matrix-seventh stage 61](https://livenorthumbriaac-my.sharepoint.com/personal/w18017575_northumbria_ac_uk/Documents/Desktop/Dissertation2.0.docx#_Toc80667010)

[Figure 122.MobileNetV2-Confusion matrix-seventh stage 61](https://livenorthumbriaac-my.sharepoint.com/personal/w18017575_northumbria_ac_uk/Documents/Desktop/Dissertation2.0.docx#_Toc80667011)

[Figure 123.ResNet101V2-Confusion matrix-seventh stage 61](https://livenorthumbriaac-my.sharepoint.com/personal/w18017575_northumbria_ac_uk/Documents/Desktop/Dissertation2.0.docx#_Toc80667012)

[Figure 124.VGG19-Confusion matrix-seventh stage 61](https://livenorthumbriaac-my.sharepoint.com/personal/w18017575_northumbria_ac_uk/Documents/Desktop/Dissertation2.0.docx#_Toc80667013)

[Figure 125.MobileNetvV2-Clasification report-seventh stage 62](https://livenorthumbriaac-my.sharepoint.com/personal/w18017575_northumbria_ac_uk/Documents/Desktop/Dissertation2.0.docx#_Toc80667014)

[Figure 126.InceptionV3-Clasification report-seventh stage 62](https://livenorthumbriaac-my.sharepoint.com/personal/w18017575_northumbria_ac_uk/Documents/Desktop/Dissertation2.0.docx#_Toc80667015)

[Figure 127.ResNet101V2-Clasification report-seventh stage 62](https://livenorthumbriaac-my.sharepoint.com/personal/w18017575_northumbria_ac_uk/Documents/Desktop/Dissertation2.0.docx#_Toc80667016)

[Figure 128.VGG19-Clasification report-seventh stage 62](https://livenorthumbriaac-my.sharepoint.com/personal/w18017575_northumbria_ac_uk/Documents/Desktop/Dissertation2.0.docx#_Toc80667017)

[Figure 129.Training results for ResNet101V2 in the eighth stage 63](https://livenorthumbriaac-my.sharepoint.com/personal/w18017575_northumbria_ac_uk/Documents/Desktop/Dissertation2.0.docx#_Toc80667018)

[Figure 130.ResNet101V2-eighth stage 63](https://livenorthumbriaac-my.sharepoint.com/personal/w18017575_northumbria_ac_uk/Documents/Desktop/Dissertation2.0.docx#_Toc80667019)

[Figure 131.ResNet101V2-Confusion matrix-eight stage 64](https://livenorthumbriaac-my.sharepoint.com/personal/w18017575_northumbria_ac_uk/Documents/Desktop/Dissertation2.0.docx#_Toc80667020)

[Figure 132.ResNet101V2-Clasifiation report-eighth stage 64](https://livenorthumbriaac-my.sharepoint.com/personal/w18017575_northumbria_ac_uk/Documents/Desktop/Dissertation2.0.docx#_Toc80667021)

# List of tables:

[Table 1.Facial emotion recognition datasets 32](#_Toc80554526)

[Table 2.Number of samples in each class 33](#_Toc80554527)

[Table 3.ResNet101V2 architecture 34](#_Toc80554528)

[Table 4.VGG19 architecture 34](#_Toc80554529)

[Table 5.InceptionV3 architecture 35](#_Toc80554530)

[Table 6.MobileNetV2 architecture 36](#_Toc80554531)

[Table 7.Models comparison 67](#_Toc80554532)

# Introduction

## 1.1. Motivation

Every human being has had the experience of expressing emotions as a form of communication in some manner. Individuals convey their emotions in a variety of ways, including via body language, tone of voice, and facial expression. (Routray & Happy, 2015). (Majumder et al., 2018) .

Academics and philosophers have long been fascinated by the recognition of facial expressions. In 1872, Charles Darwin suggested six universal emotions in 1872, which everyone everywhere can read and understand. (2011) (D. Matsumoto &, H. S. Hwang, 2011). Numerous academics have disputed this claim, including Paul Ekman, Wallace Friesen, and Phoebe Ellsworth (D. Matsumoto & H. S. , D., Hwang, H.S., 2011). Silvan Tomkins later recruited Paul Ekman and Carroll Izard to research Charles Darwin’'s hypothesis, and they discovered that people can express and understand the following emotions regardless of their location or culture: surprise, fear, disgust, anger, happiness, and sadness. (Routray & Happy, 2015; Luettin &, Fasel, 2019).

Ekman demonstrated that by analysing facial expressions, it is possible to infer patterns that can be used to anticipate behaviours or thoughts (P. Ekman & F. Friesen, 1984). Today, technology advances have created new possibilities for detecting facial expressions which have a wide range of applications in artificial intelligence. In robotics, facial expressions are critical for human-computer interaction. For a robot to be capable of mimicking human behaviour, it must first self-learn to link inner emotions with facial movements (Zhang et al., 2013).

Teaching has changed significantly in recent years, and as a result of the pandemic, e-learning has surpassed traditional classroom teaching as the most popular form of instruction. While facial expression recognition (FER) technology may assist instructors in assessing students’ learning levels, it can also predict students’ enjoyment and understanding during a lecture session (Lansley, 2020; Gu & Wang, 2018). Additionally, physicians have begun to utilise FER to treat mental and behavioural disorders (Samadiani et al., 2019). The technology may combine facial expression detection, body recognition, and speech tone identification to provide additional information to assist physicians in making judgments about the emotional and mental health of individuals diagnosed with autism, depression, or schizophrenia, among other conditions (Lansley, 2020; Gu & Wang, 2018).

Large businesses are increasing their sales by using FER. They capture people’s emotions of amazement when confronted with a novel item such as a phone or a car. They may use this approach to obtain an honest answer while presenting the design and specifications of the product and forecasting its profitability. Additionally, they want to develop a new way of recruiting trustworthy employees. This method involves recording potential employees’ answers to a questionnaire and then analysing the employees using facial expression detection (Gu & Wang, 2018). Airports, the military, and security firms are also interested in researching the use of cameras and software to combine existing knowledge about facial emotion recognition and deception in order to detect suspicious movements indicative of drug trafficking, terrorist attacks, or any other type of crime that might occur in a large and busy space (Lansley, 2020; Wang et al., 2018).This research's primary goal is to create a deep learning model for categorising face images according to their emotional state.

This research’s primary goal is to create a deep learning model for categorising face images according to their emotional state. Over the years, many researchers have tried to develop their own hypotheses about how emotions may be detected on the face. Apart from the actual code for facial emotion recognition, further tests were conducted to ascertain the performance of models at different stages of development and how the best models fared compared to some similar programmes. The comparison involves classification, training, and testing processes. Various deep convolutional neural network (CNN) architectures were used, including pooling layers, convolutional layers, and fully connected layers. These designs have been refined to improve their suitability for the task of facial emotion recognition.

## 1.2. Aims

* Create a deep learning software capable of autonomously interpreting the emotions displayed on a human face utilising a challenging dataset.
* Optimise the program’s performance by using available resources to generate an accurate model.
* Compare the optimal version to other methods provided by a variety of researchers.

## 1.3. Objectives

1. Investigate suitable tools, techniques, and methods to complete the project
2. Find the most suitable methods for building the program
   1. Find suitable pre-trained models
   2. Find libraries and software needed to develop the algorithm
3. Train the program to detect and recognise facial emotion from the pictures
4. Tune the parameters to increase the accuracy
5. Evaluate the best model in parallel with the work done by other researchers
   1. Discuss the results and highlight the good practice
   2. Discuss the flows of the system

## 1.3. Structure of the dissertation

Chapter 1 discusses the rationale and context for developing a facial emotion detection system. Additionally, it defines the dissertation’s goals and objectives as well as the dissertation’s organisation.

Chapter 2 summarises several research articles describing the methods used by various research groups to develop an appropriate model for facial emotion detection.

Chapter 3 discusses the Deep Convolutional Neural Network’s (DCNN) features and components, how these components interact, and the critical values that must be considered while building this architecture.

Chapter 4 details the selection of the dataset, pre-trained models, tools, and libraries. Each of these components has been explained in depth to ensure that they are well understood.

Chapter 5 outlines the whole process of training and testing models. It explains the changes made at each step in response to the obtained findings and suggestions from other researchers in order to avoid the issue of underfitting or overfitting.

Chapter 6 compares the final model to various techniques developed by academics using the same dataset. Additionally, it provides a summary of the accomplishments made throughout the project’s development. It also discusses the unanticipated difficulties faced and how they were resolved.

Finally, Chapter 7 is about the recommendations that can be made to improve the performance of the algorithm developed for facial emotion recognition. It also includes a summary of all the work done in this study and the results achieved.

# Analysis

# Chapter 2. Related work

Mollahosseini et al. (2016) suggest the use of deep CNNs for facial emotion recognition across a variety of publicly accessible datasets. The pictures from the dataset were resized to 48x48 pixels after facial landmarks were extracted from the data. The researchers then used the method of data augmentation. Two convolution-pooling layers were utilised in the architecture followed by two inception-type modules including convolutional layers of sizes 1x1, 3x3, and 5x5. They demonstrate the ability to utilise the network-in-network method, which enables increased local performance due to the locally applied convolution layers and reduces the overfitting issue.

(Lopes et al., 2010) investigated the effect of pre-processing data before training the network to improve emotion categorisation. Before CNN, data augmentation, rotation correction, cropping, downsampling with 32x32 pixels, and normalisation were performed. CNN consists of two convolution-pooling layers that terminate in two dense layers with 256 and 7 neurons, respectively. At the test stage, the best weight acquired throughout the training stage is utilised. Three publicly available datasets were used to assess this experience: CK+, JAFFE, and BU-3DFE. According to Lopes et al., using all of these pre-processing stages together is more successful than doing them individually. Additionally, Mohammadpour et al. (2017) have used these pre-processing methods. They suggest a new CNN for the detection of facial AUs. They have utilised two convolution layers for the network followed by a max-pooling layer for each and two fully connected layers to show the number of active AUs.

In 2018, Cai et al. proposed a new architecture CNN with sparse batch normalisation for the vanishing or explosion gradient issue. The characteristic of this network is that it employs two convolutional layers sequentially followed by max pooling and SBP and, to mitigate the overfitting issue, a dropout is used throughout three fully connected layers. For the face occlusion issue, Li et al. (2019) propose a novel CNN approach in which they first input data into a VGGNet network and then use the CNN technique utilising the attention mechanism ACNN. Three extensive databases were used to train and test this architecture: FED-RO, RAF-DB, and AffectNet.

(Yolcu et al., 2019) developed a method for detecting a person’s fundamental facial features. To achieve their results, they used three CNNs with the same architecture, each of which detected a different face region, such as the brow, the eye, and the mouth. Cropping and identification of key-point facial features were conducted before the pictures were introduced into the CNN network. Using the iconic face developed in conjunction with the raw picture, the second kind of CNN was created to identify facial emotion. Researchers have shown that this approach outperforms the use of raw pictures or iconised faces alone in terms of accuracy.

(Agrawal and Mittal, 2019) researched the effect of change in CNN parameters on recognition rate using the FER2013 dataset, and the results are published online. First and foremost, all pictures are specified at 64x64 pixels which differ in size and the number of filters used. Includes the kind of optimiser used (Adam, stochastic gradient descent [SGD], Adadelta) on a basic CNN, which has two consecutive convolution layers. The second layer serves as a max-pooling layer followed by a softmax function for classification on the final layer. The researchers developed two new CNN models with average accuracy of 65.23% and 65.77%, respectively. The distinctive feature of these architectures is that they do not include completely connected layers that drop out, and the same filter size is maintained throughout the network.

According to (Jain et al., 2019), a new deep CNN with two residual blocks each containing a four-convolution layer has been proposed. After the pre-processing phase, which allows the cropping and normalisation of the intensity of the pictures, this model is trained on the JAFFE and CK+ databases. Kim et al. (2020) investigated the variation in facial expressions associated with changes in emotional state. They suggested a spatial-temporal architect that is a mixture of CNN and LSTM to study facial expression variations. After learning the spatial characteristics of the facial expression in all the frames of the emotional state, a CNN was used to preserve the whole sequence of these spatial features. After that, an LSTM was used to maintain the entire sequence of these spatial features. Yu et al. (2018) have also contributed to this work and have proposed a novel architecture known as spatial-temporal convolutional features with nested LSTM. This architecture is based on three deep learning sub-networks: 3DCNN for extraction of spatial-temporal features, temporal T-LSTM to preserve the temporal dynamics, and the convolutional C-LSTM for modelling the multi-level features, among others.

(Liang et al., 2020) suggested a deep convolutional BiLSTM architecture in which they create two DCNNs, one of which is designated for obtaining spatial features from facial expression sequences and the other for obtaining temporal features from facial expression sequences. These features are fused at the level of a vector with 256 sizes, and for classification into one of the six regular emotions, researchers used a BiLTSM network. When it came to the pre-processing step, they utilised the multitask cascade convolutional network for identifying faces, and then they used data augmentation to expand the dataset.

(Schoneveld et al., 2021) presented a novel deep learning-based method for audio-visual sentiment detection in this paper. This method uses current advancements in deep learning such as knowledge distillation and high-performance architectures. A model-level fusion approach is utilised to merge the deep feature representations of the auditory and photographic modalities. The temporal dynamics are then captured using a recurrent neural network. On the RECOLA dataset, the suggested method significantly surpasses state-of-the-art algorithms in predicting valence. Furthermore, using the AffectNet and Google facial expression comparison databases, the proposed visual facial expression feature extraction network surpasses state-of-the-art findings.

While convolution padding aids in the collection of edge information, it also erodes the feature map. Convolution with many layers results in an output feature map called the albino feature, which significantly degrades the representation of the expression. To address these issues, (Shi et al., 2021) offer an innovative design called the amending representation module (ARM). ARM serves as a replacement for the pooling layer. In theory, it may be integrated into the back end of any network to address padding erosion. ARM effectively improves facial expression representation in two ways: by decreasing the weight of eroded features to compensate for the padding effect and by sharing affinity features across mini-batches to boost representation learning.

(Savchenko, 2021) examined the multi-task learning (MTL) capabilities of lightweight CNNs for face recognition and classification using cropped faces without borders. Savchenko emphasises the need to fine-tune these networks for facial expression prediction. Numerous models based on the MobileNet, EfficientNet, and ResNet designs are provided. They were shown experimentally to provide near-state-of-the-art results for age, gender, and race identification on the UTKFace dataset as well as emotion classification on the AffectNet dataset.

(Vo et al.,2020) discussed automated FER on a single in-the-wild (ITW) picture. ITW pictures have significant posture, direction, and input resolution issues. This research solved the ITW FER problem using a pyramid with a super-resolution network design. Additionally, a previous distribution label smoothing loss function was added, which applies extra prior information about the confusion associated with each phrase in the FER task. Experiments on three of the most prominent ITW FER datasets demonstrated that their technique surpasses all existing approaches.

(Pourmirzaei et al., 2021) examined the effect of ImageNet pre-training on FER at various augmentation degrees. As a consequence of the findings, it can be concluded that training from scratch achieves superior performance to ImageNet fine-tuning at higher augmentation levels. Following this, a framework for conventional SL was developed and termed hybrid learning. This framework combines self-supervised co-training with SL consistent with MTL. Self-supervised learning obtained additional information from input data, such as spatial information from faces, which aided the primary SL job. The feasibility of using this technique for FER issues has been explored using self-supervised pre-tasks such as jigsaw puzzles and in-painting. These two approaches aided the supervised head in lowering the error rate when various augmentations and low data regimes were used in the same training conditions. On AffectNet, the state of the art was achieved using two distinct hybrid learning techniques without different datasets. Additionally, the impact of hybrid learning was shown on two distinct facial recognition problems—head position estimation and gender identification—which resulted in an error rate reduction of up to 9% and 1%, respectively. Additionally, they observed that the hybrid learning techniques prevented the model from overfitting.

(Schoneveld et al., 2021) provide a novel method of audio-visual emotion detection based on deep learning in this article. This method capitalises on recent advancements in deep learning, such as knowledge distillation and high-performance deep architectures. A model-level fusion approach is utilised to combine the deep feature representations of the auditory and visual modalities. To capture the temporal dynamics, a recurrent neural network is then utilised. On the RECOLA dataset, our suggested method significantly outperforms existing algorithms in predicting valence. Furthermore, the proposed network for extracting visual facial expression features surpasses state-of-the-art findings on the AffectNet and Google facial expression comparison datasets.

(Siqueira et al., 2020) present studies on ensembles with shared representations based on convolutional networks to show their data processing efficiency and scalability to large-scale facial expression datasets statistically and qualitatively. They demonstrate that by changing the branching level of the ensembles with shared representations, it is possible to significantly decrease redundancy and computational burden without sacrificing variety or generalisation power, which are critical for ensemble performance. Experiments on large-scale datasets indicate that ensembles with shared representations significantly reduce the remaining residual generalisation error on the AffectNet and FER+ datasets, achieve human-level performance, and outperform state-of-the-art methods for recognising facial expressions ITW that incorporate emotion and affect concepts.

The review conducted by (Ko, 2018) concentrated solely on studies that utilise facial pictures since visual expressions are one of the primary conduits of information in interpersonal communication. This article summarises decades of study in FER. It presents traditional FER methods followed by a description of typical FER system types and their primary algorithms. The authors next propose FER methods based on deep learning that use deep networks to enable end-to-end learning. Additionally, this study discusses a cutting-edge hybrid deep-learning method that employs a CNN for the spatial characteristics of an individual frame and a long short-term memory for the temporal characteristics of successive frames. The last section provides a brief overview of publicly accessible assessment criteria and a comparison to benchmark findings, which serve as a baseline for quantitative comparisons of facial emotion recognition research.

# Chapter 3. Literature review

## 3.1 General approach to facial emotion recognition

The development of the facial emotion recognition system generally consists of five different stages: image acquisition, pre-processing, feature extraction, classification, and post-processing. (Chibelushi and Bourel,2016)

Figure 1.The facial emotion recognition process

Image acquisition refers to the process of locating pictures, which may be static photos or image sequences depicting faces from various angles. Both kinds of pictures may be two dimensional or three dimensional, but image sequences offer additional information by describing the temporal aspects of an expression (Chibelushi & Bourel, 2016). There are some common issues regarding all the object recognition tasks, such as illumination, deformation, segmentation, viewpoints, and affordance. Illumination depends on the source of light, which can change the intensity of each pixel. Deformation can occur when an object is manually modified, such as a hand-written object that can be represented differently depending on each person. Segmentation problems are due to real-life pictures being cluttered with numerous objects, making the detection more complicated. The viewing angle can be seen from different perspectives, in which case learning methods cannot handle. Affordance describes the relationship between the object and its purpose. For example, an object can have different shapes and colours, but it has the same functionality (Prasad, 2012).

Pre-processing is used to address the problems previously stated. Geometric normalisation enables the modification of head translation, rotation, and scale. Additionally, picture segmentation may be enhanced by including other models such as a Gaussian mixture model of the skin or deformable models of the facial features. Furthermore, the illumination problem can be solved using different tools and techniques, such as histogram equalisation (Chibelushi & Bourel, 2016).

According to Lv, Feng, and Xu (2014), face detection is a critical job in such a challenge, and there are many methods for this: appearance-based approaches, template-based approaches, feature-based approaches, and local-global graph approaches.

* The appearance-based method involves training the classifier on patterns with and without faces, with the face being recognised as a whole. The advantage of this technique is that it can achieve excellent accuracy with just frontal pictures that are brightly lit and have a clean backdrop.
* To identify a face using the template-based method, a generalised face shape is created and linked with a picture. This job requires generalising various shapes, sizes, and poses, which may be challenging.
* The feature approach is based on detecting features that do not change over time. Therefore, it could be difficult to find this feature in an overloaded background with low light.
* The local-global approach compares faces using a graph with nodes according to similarities in colour, translation, rotation, and scale. This can be difficult if the pictures are low quality and small.

The objective of the feature extraction step is to acquire stable and discriminative face characteristics from two viewpoints. The goal is to derive these characteristics using geometric feature-based or appearance-based techniques. If the geometric feature approach is employed, links between face regions in terms of position and form must be established, which has the drawback of requiring sophisticated and reliable detection methods. The appearance-based methods only apply overlapped filters on the face or over a section of the face. Scientists have been using principal component analysis, independent component analysis, and Gabor wavelets showing varying degrees of success (Chibelushi & Bourel, 2016; Lv, Feng, & Xu, 2014; Liu et al., 2014; Fasel & Luettin, 2003.)

In the classification part, different classifiers can be used, such as Hidden Markov models, k-nearest neighbours, support vector machines (SVMs), et cetera. This shows to which class a picture belongs using action units or prototypic facial expressions (Chibelushi & Bourel, 2016).

## 3.2 Classification

A model that classifies facial expressions requires two integral components: loss function and score function. The loss function helps estimate how close the prediction is to a true value, while the score function maps the data to class scores. Any model needs an optimisation that ensures a minimum loss function without affecting the score function’s parameters (Ufldl.stanford.edu, 2021; Cs231n.github.io, 2021).

### 3.2.1 Description of linear classifiers

Linear classifiers are functions that generate a linear mapping of the input data. These are constructed via the use of linear combinations of a set non-linear basis function. The equation describing the linear classifier is as follows (Bishop, 2006):

where f(\*) is the identity for the regression problems and the non-linear activation function for classification problems (Bishop, 2006). w is created by the weight of coefficients and biases, and 𝜙𝑗(𝒙) is a non-linear function that depends on parameters. A linear classifier is simple to use and understand, but it has a substantial limitation that does not allow for the use of data in more than two classes.

### 3.2.2 Non-linear classifiers

To overcome the limitation of a linear classifier, models such as neural networks were developed that can learn non-linear features.

## 3.3 Neural networks

Neural networks were inspired by the human biological brain and allow computer programs to solve commune problems in the field of AI. A neural network is a collection of functional transformations conducted by computing units referred to as neurons. Neural networks employ a W vector with flexible parameters to control the input and the output set, which are mainly used to obtain a non-linear function since these specific networks operate with essential functions (Bishop, 2006; Cs231n.github.io, 2021). By comparison, the SVM with the same generalisation performance as the neural network model is less compact and challenging to evaluate (Bishop, 2006). Due to the high compactness of the model, the complexity of the function increases.

### 3.3.1 Feed-forward neural network

A feed-forward neural network is an artificial neural network where the information goes only in one direction starting from the input nodes through the hidden nodes to the output nodes without forming any connection in a circle. The equation representing this neural network is as follows:

This is made using M combinations of the input variables x1,x2…xN where I = 1,…,M; wj0(1) points the bias, wji(1) the weights, and aj the activation (Bishop, 2006). This activation is created by applying activation function h(\*), which develops the following:

To be able to learn non-linear combinations from the original data, the feature extraction procedure is performed by the output quantities of the function mentioned above, which are called hidden units or hidden neurons. This process is valuable for issues where the original input features are not exceptionally independently informative. (Murphy, 2012).

The basis function correlates with the activation function, which in many cases is a sigmoidal function. These functions are known as the tanh function, rectified linear units function, and logistic sigmoid. As previously stated, these activation units define layers that are categorised as input, hidden, and output. For the output neurons placed in the last layer, the activation function is chosen depending on the data and the number of target variables (Bishop, 2006). For a binary classification problem, a logistic sigmoid is used. If the target number of classes is more than one, then a softmax function is used. As for regression problems, an identity function is chosen (Murphy, 2012; Bishop, 2006).

The output of each hidden layer combined produces the following output unit activation:

Where k =1,…, N and N represents the overall number of outputs, ak  is the output neuron activations, wk0(1) are the parameters, and wkj(1) are the weights.

The neural network function as a whole can be represented by combining the previous equations as follows:

Where f and g are non-linear functions and w is the vector containing all the weight and parameters combined.

Creating a supervised neural network means creating an algorithm that can learn to build a model that represents relations between the input data and the goal variables. Therefore, the model becomes more accurate as the output values get closer to the target value. This is often connected to selecting the activation and error functions that correspond to the kind of issue being addressed. A multi-class cross-entropy error function is utilised in the case of the softmax classifier. As for binary classification, cross-entropy and logistic sigmoid errors are used. In the case of regression, sum of a square and linear outputs are the choice. These objective functions are used to evaluate the model by determining which weights have the lowest value (MacKay, 2003; Murphy, 2012; Bishop, 2006).

## 3.4. Activation function

An activation function is a mathematical operation that defines the output of that unit given an input or a set of inputs. The most common activation functions are maxout, sigmoid, tanh, and ReLu.

### 3.4.1. Tanh

In this activation function, the values vary between the range of [-1, 1], and its equation is as follows:

where e represents the Euler’s number and x is the input number. The problem is that it saturates the gradients.

### 3.4.2. Sigmoid

The sigmoid function takes values between [0, 1] and has the following mathematical equation:

where e represents the Euler’s number and x is the input number. The disadvantages of using this kind of activation function are that it tends to vanish the gradient and the sigmoid outputs are not zero-centred ( Cs231n.github.io., 2021).

### 3.4.3. ReLU and Leaky ReLU

In contrast to the Sigmoid and Tanh activation functions, this one does not constrain the input to a range but restricts it to 0 (Poczos & Singh, 2016; Cs231n.github.io., 2021; Goodfellow et al., 2016). It also speeds up the gradient descendent optimisation algorithm and is much simpler to implement. The equation describing this activation function is as follows:

The disadvantage with these activation functions is that the neurons within layers could become inactive at high learning rates. An alternative activation function called Leaky ReLU was developed to fix this issue. This equation is defined as follows

### 3.4.4.Maxout

This type of activation function is beneficial, especially for training with dropout, and able to train larger neural networks. It has all the advantages of the ReLU without its disadvantages (Goodfellow et al., 2013; Cs231n.github.io., 2021). The equation that defines this activation function is as follows:

The drawback of this activation function is that the number of parameters doubles in size (Cs231n.github.io., 2021).

## 3.5 Gradient descendent optimisation

Gradient descendent optimisation is utilised during a neural network training phase to determine which vector w provides the minor value E(w). That lowest regard happens at a position in the weight space where the error function’s gradient vanishes:

These points are called stationary points, and they are divided into minima, maxima, and saddle points. In some cases, many of these points may have the same value. For minima points, the equal points receive the name local minima. Moreover, the smallest number receives the name global minimum (Bishop, 2006).

Bishop (2006) states that comparing multiple local minima can be enough to find a suitable solution, and it may not be necessary to obtain the global minimum. Most approaches entail choosing initial weight values and then iteratively travelling through the weight space in subsequent steps to locate the answer for ∇𝐸(𝒘) = 0. The mathematical form is as follows:

Where 𝜏 is the iteration step.

Gradient information can be used to conduct the changes. This data is utilised to enhance the speed with which the weight vector that generates a satisfactory solution is found. Batch, minibatch, and online methods are the three most common methods for this update. The batch technique processes the entire dataset at once, whereas the minibatch approach processes only a portion of it, and the online method processes one data point at a time.

The strategy is known as gradient descent or steepest descent when the batch method is utilised, and its mathematical form is as follows:

Gradient descent is less resilient and faster for batch optimisation than conjugate gradient and quasi-Newton techniques. The benefit of these approaches over gradient descent is that the error function always decreases with each repetition. This only happens if the weight vector is not at a local or global minimum (Bishop, 2006; Burlutskiy et al.,2016). The methodology is known as mini-batch gradient descent when the mini-batch method is utilised. As previously stated, the distinction is that the mini-batch utilises N examples instead of the entire set in each iteration (Liu et al., 2019).

When the online approach is utilised, the last methodology is recognised as sequential gradient descent or SGD. This technique updates the weight vector with just one data point at a time either by cycling through the information in order or by randomly picking points with replacement (Bishop, 2006; Liu et al., 2019). The equation is defined as follows:

Compared to batch and mini-batch gradient descent, SGD offers two significant advantages: SGD is more efficient in dealing with data redundancy and can avoid local minima.

### 3.5.1 Gradient descendent optimisation algorithms

To prevent being stuck in suboptimal places, gradient descent optimisation techniques were created. Momentum, Nesterov’s accelerated gradient, Adagrad, Adadelta, RMSprop, and Adam are among the most widely used methods to tackle this problem.

#### 3.5.1.1 Momentum

T This approach aids gradient descent by decreasing the influence of ravines, which are regions where the surface bends more sharply in one dimension than the other and are prevalent around local minima (Ruder,2016). To mitigate this impact, the technique multiplies the current update by a portion of the previous update, which is expressed as follows:

where momentum 𝜇 ∈ [0, 1] is the weight of the previous update and 𝑣 𝜏+1 and 𝑣 𝜏 are the current and previous iterations’ updated values, respectively.

When the updates travel in the same direction, the updated value grows, and when the updates move in opposite ways, the updated value drops. As a result, faster convergence and less oscillation are achieved (Ruder, 2016).

#### 3.5.1.2 Nesterov Accelerated Gradient

The Nesterov accelerated gradient technique is superior to the momentum technique because it considers the gradient’s potential value and performs a rectification before leaping to the value (Ruder, 2016; Cs231n.github.io., 2021). Because the “lookahead” gradient step regularly performs somewhat better than regular momentum in practice, the equation is expressed as follows:

where ∇𝐸(𝑤𝜏 − 𝜇𝑣𝜏 ) is the “lookahead” gradient step.

#### 3.5.1.3 Adagrad

An adaptive learning rate approach changes each parameter’s dynamic learning rate by updating infrequent parameters more quickly and updating frequent parameters at a slower rate. The progress of uncommon and frequent parameters evens out over time, which is an excellent feature (Keras, 2021).

Where 𝜖 is a smoothing term

Adagrad has the benefit of automatically adjusting the learning rate. As a result, the learning rate utilised in implementation is typically 0.01. (Ruder,2016) There are a few disadvantages to Adagrad. The first disadvantage is that this approach is very dependent on the learning rate, which becomes low when the starting gradient is significant. The second drawback is that the monotonic learning rate is excessively aggressive due to the L2 normalisation of all prior parameters computed in the denominator. (Zeiler,2017), (Ruder,2016), (Cs231n.github.io., 2021)

#### 3.5.1.4 Adadelta

This technique is similar to Adagrad, but it attempts to minimise the forceful monotonic learning rate by limiting the number of collected previous gradients to a set number determined by a window of size 𝜔. Adadelta does not retain prior squared gradients inefficiently to define the number of gradients across a window. The technique instead uses a declining average of all 𝜔 previously squared gradients. The prior average and the current gradient are used to calculate this decaying running average (Keras, 2021). As a mathematical equation, it is defined as follows:

where 𝐷[ϕ]𝜏 is the running average at iteration 𝜏, 𝜌 is a constant that controls the decay of the previous parameter updates, and Δ𝑤 refers to the previous weight update.

Adadelta has the benefit of addressing Adagrad’s shortcomings. Furthermore, it eliminates the need to modify the hyper-parameters despite a wide range of input data formats, hidden unit numbers, and non-linearities. Adadelta is therefore a stable learning rate approach (Ruder, 2016).

#### 3.5.1.5 RMSprop

Hinton (2016) offered an unpublished adaptive learning rate technique as a new way to improve Adagrad. RMSprop modulates the learning rate of each weight by using a moving average of squared gradients (Keras, 2021; Cs231n.github.io., 2021). This optimisation is represented as follows:

where is the running average of squared gradients at repetition 𝜏 and 𝐷𝑅 (decay rate) is a hyper-parameter, whose typical value is 0.9, 0.99, or 0.999 (Zeiler, 2017; Hinton, 2016).

RMSprop addresses the vanishing learning rate produced by monotonically more minor updates in the same manner as Adadelta does. (Cs231n.github.io., 2021)

#### 3.5.1.6 Adam

The adaptive learning rates for each parameter are calculated using adaptive moment estimation (Adam), an alternate technique (Keras, 2021). By tracking both an exponentially decaying average of previous square gradients 𝑣𝜏 and an exponentially decaying average of previous gradients 𝑚𝜏, Adam combines the ability of Adagrad to cope with sparse gradients with the capability of RMSprop to cope with non-stationary goals (Kingma & Ba, 2015). The first element corresponds to the gradients’ second instant (variance). The second element signifies the initial instant of the gradients (mean). 𝑣𝜏 and 𝑚𝜏 values correspond to 𝐷[ϕ 2 ]𝜏 and 𝐷[ϕ]𝜏 (Ruder, 2016).

𝑣𝜏 and 𝑚𝜏 are initialised as vectors of 0 as stated in Kingma and Ba’s (2015) work. The moment estimations are therefore skewed towards 0, especially when the decay rates are modest, and the period steps are short. They do it in the following ways to offset these biases:

where 𝛽1 and 𝛽2 are the moment estimates’ exponential decay rates and 𝑚̂𝜏 and 𝑣̂𝜏 are the bias-corrected first and second raw moment estimates, respectively.

. Adam uses very modest memory, has better performance than adaptive learning approaches across a wide range of models and datasets, and is scalable to large-scale, high-dimensional machine learning challenges (Kingma & Ba, 2015).

### 3.5.2 Backpropagation

The derivatives of the error function concerning the weights may be evaluated using backpropagation, a computationally efficient approach (Bishop, 2006; Goodfellow et al., 2016). This technique is intended to determine how much of a node’s error contribution is in the output. This algorithm is described by Rodrigues (2015), Bishop (2006), and Goodfellow et al. (2016) as follows:

* "Apply a feed-forward to an input vector 𝑥𝑛 using 𝑎𝑗 = ∑𝑗=0 𝑤𝑗𝑖𝑥𝑖 and 𝑧𝑗 = ℎ(𝑎𝑗) to find the activations of all the hidden and output units.
* Evaluate 𝛿𝑘 for all the output units using 𝛿𝑘 = 𝑦𝑘 − 𝑡𝑘.
* Backpropagate the 𝛿𝑘 using 𝛿𝑘 = ℎ′(𝑎𝑗) ∑𝑘 𝑤𝑘𝑗𝛿𝑘 to obtain 𝛿𝑘 for each remote unit in the network.
* Use 𝜕𝐸𝑛 /𝜕𝑤𝑖𝑗 = 𝛿𝑗𝑧𝑖 to evaluate the required derivatives."

## 3.4 Challenges of building a model

It is not easy to create a machine learning model. This section discusses common issues in all models as well as specific limitations when employing gradient descent that must be taken into account during the training and evaluation process.

### 3.4.1 Description of Bias-Variance compromise, overfitting and underfitting

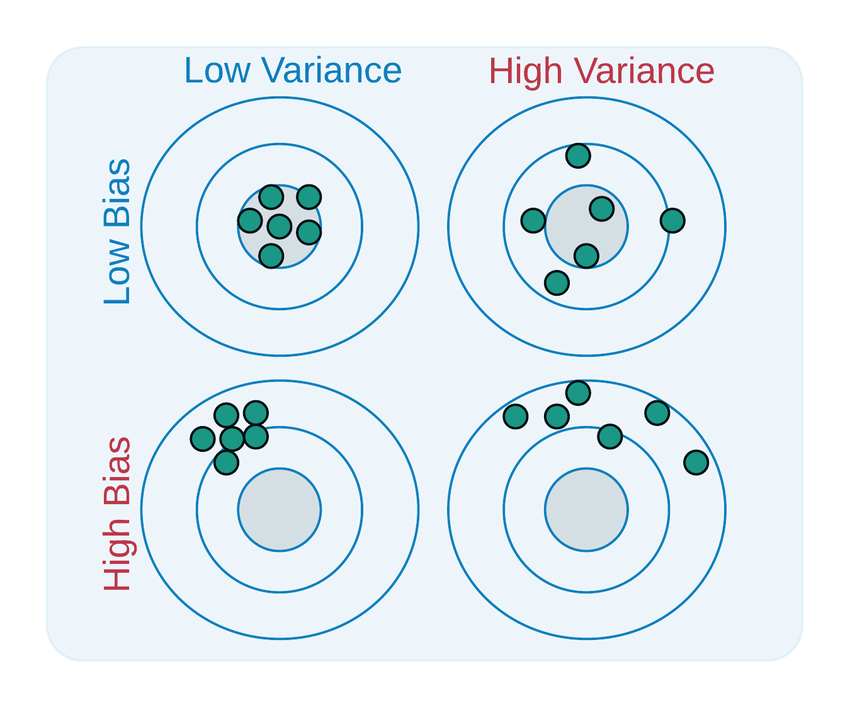
 It is crucial to remember that data modelling aims not to develop a model that precisely matches the training data but to develop one that can generalise new information. The notions of bias and variance are strongly connected to this objective. It is crucial to remember that data modelling aims not to develop a model that precisely matches the training data but to develop one that can generalise new information. The notions of bias and variance are strongly connected to this objective. The first element of machine learning, bias, relates to the difference between the expected and actual values; the second element, variance, refers to the target’s fluctuation around its true mean (Hastie et al., 2001; Goodfellow et al., 2016). This is apparent in Figure 2.

Figure 2.Bias-variance compromise

The model is overfitting the data when it has a low bias and a significant variance. This implies that the projected values roughly match the actual values (the green spots are close to the grey objective), but the inconsistency of these predictions between trials is significant (the green spots are dispersed). On the other hand, if the model has a significant bias and low variance, it is underfitting the data. This implies that while the difference between anticipated and actual values is considerable (the green spots are far from the grey goal), the variability of the predictions between trials is minimal (the green spots are gathered). Overfitting can be avoided by lowering the number of elements in the parameter space or decreasing the adequate size of each element.

Pruning and weight sharing are two strategies used to minimise the number of dimensions in the parameters. Pruning is the process of eliminating information from an overfitted model. The procedure varies depending on the model, but the concept remains the same. For example, pruning a decision tree model involves replacing the subtrees with leaves, while pruning a neural network means that the irrelevant weights are eliminated. Weight sharing is a way of distributing a single weight over several connections in a network. This indicates that the network’s number of adjustable weights is fewer than the number of connections (Hinton, 2016; Hastie et al., 2001; Cun et al., 1990).

Regularisation and early stopping are the strategies used to minimise the size of each dimension. Regularisation is a technique for reducing overfitting by penalising the loss function for its complexity. Early stopping is a way of halting the training process as soon as the model’s performance stops improving or declines (Begio, 1998; Hastie et al., 2001; Goodfellow et al., 2016).

### 3.4.2 Disappearing gradient and inflating gradient

When deep neural networks and neural networks are trained using gradient descent, two major issues arise: disappearing gradients and inflating gradients. The first issue arises when the gradient signal weakens in previously hidden layers as it travels back through several layers, leading the learning process to become trapped in bad local minima. In other words, neurons in the later layers of the brain learn more quickly than neurons in the early levels do. The second issue arises when neurons in later layers learn more slowly than neurons in earlier levels do (Murphy, 2012).

Unsupervised learning, also known as generative pre-training, can be used to prevent these issues by initialising the parameters. Unsupervised learning has the benefit of forcing the model to characterise a high-dimensional answer, such as the input feature vector, instead of expecting a scalar answer. This works as a data-induced regulariser, assisting backpropagation in discovering local minima with strong generalisation characteristics (Murphy, 2012).

Changing the activation function is another method to avoid this. In the past, the sigmoid function was the chosen function; however, owing to the drawbacks stated, this trend has shifted. As previously noted, the sigmoid function’s alternate activation functions include tanh, ReLU, and maxout. Tanh eliminates the zero-centred nuisance, yet it continues to suffer from vanishing and increasing gradients. ReLU has one major flaw: when the learning rate is too high, the network’s units die during training. Finally, maxout appears to be the most reliable choice since it generalises ReLU and leaky ReLU and provides the benefits of ReLU without the downsides (Cs231n.github.io., 2021; Goodfellow et al.,2016).

## 3.5 Convolutional Neural Network using Deep Learning

CNNs, a kind of multilayer perceptron, are neural networks specifically intended to exploit data structure, such as 1d signals like voice or text or 2d signals like pictures (Murphy, 2012; Bishop, 2006). The primary distinction between CNNs and DCNNs is that DCNNs have a higher number of hidden layers than CNNs. Convolutional layers, pooling or subsampling layers, and fully connected layers are used in both architectures (Goodfellow et al., 2016).

### 3.5.1 Convolutional layer

The convolutional layer is a crucial component of the network since it converts one activation capacity into another by convolving a tiny region with the input volume. Discrete convolution is the name for this procedure. Convolutional layers deliver to the network two essential features: local connectivity and parameter sharing. When neurons are linked to a specific part of the input volume, this is known as local connectivity. This local area is defined by the hyper-parameter receptive field, which has dimensions of d\*d. The number of connections in a neural network is substantially reduced when local connectivity is used. This not only reduces processing time but also enhances the model’s performance by minimising overfitting (Wang et al., 2016; Galeshchuk & Mukherjee, 2017).

**A collage of a person's face

Description automatically generated with medium confidence**

Figure 3.Local connectivity and receptive field

Diagram

Description automatically generated Parameter sharing happens when neural units in a group, known as a feature map or activation map, share the identical parameters and use various receptive fields to cover different picture areas. This gives the network translation invariance, which implies that beneficial characteristics learned in one part of the picture may be used in other parts of the image without having to learn them all over again (Goodfellow et al., 2016; Wang et al., 2016; Murphy, 2012).

Figure 4.Parameter sharing

There are filters that characterise sets of weights in each convolutional layer. To create an output volume, these filters, identified as kernels, are convolved with an input volume. Each convolution produces an activation map that identifies a certain feature. The number of activation maps in each layer is proportional to the number of filters in that layer (Cs231n.github.io., 2021; Wang et al., 2016; Galeshchuk & Mukherjee, 2017).

A picture containing icon

Description automatically generated Activation maps are the output of neurons after they have been convolved with the same kernel. Three hyper-parameters define their dimensions: depth, stride, and zero padding (Goodfellow et al., 2016)

Figure 5.Activation maps

The number of neural units connected to the identical part of the input volume but activated by various characteristics in the input, such as edges or blobs of colour, is controlled by depth. These neurons form a depth column (Goodfellow et al., 2016). The distance between depth columns is determined by stride. A shorter stride leads to less gap between columns and more receptive field overlap, resulting in a greater output volume. A longer stride, on the other hand, results in a smaller output volume (Goodfellow et al., 2016). Zero padding is a hyper-parameter that is utilised to pad the input volume’s boundary with zeros to regulate the output volume’s spatial size (Goodfellow et al., 2016).

Once these hyper-parameters have been established, the number of neurons that can be placed in the output volume can be calculated using the following formula:

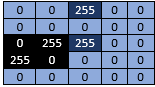
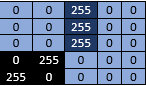
where 𝑊 is the amount of the input volume, F is the receptive field size, P is the amount of zero paddings applied on the border, and S is the stride. It is essential to keep in mind that this outcome cannot be a decimal number (Goodfellow et al., 2016; Cs231n.github.io, 2021).

Similarly, the output volume can be determined using the following formula:

### 3.5.2 Pooling or subsampling layer

The pooling layer lowers the input’s spatial size, reducing the number of parameters and amount of network processing. As a result, the network invokes local translation and a method for controlling overfitting (Cs231n.github.io, 2021). The spatial size reduction is accomplished by aggregating the activations of a collection of hidden units within a neighbourhood utilising a pooling function such as max pooling, average pooling, L2-norm pooling, or fractional max pooling (Goodfellow et al.,2016)

Figure 6.Spatial size reduction



A screenshot of a computer

Description automatically generated with medium confidencePooling can be accomplished in two different ways: overlapping and non-overlapping. With a stride of 2, the former uses a 3x3 max-pooling filter, while the latter uses a 2x2 max-pooling filter (Goodfellow et al., 2016; Cs231n.github.io, 2021).

Figure 7.Max pooling

Figure 8.Down sampling

**Calendar

Description automatically generated****A picture containing text, clock

Description automatically generated**

This convolutional layer conducts separately on each depth slice of the input and spatially changes the input's size by selecting the most considerable value within a local region using the MAX-pooling operation. The region depicted above is comprised of four pieces. The ma pooling operation can be stated mathematically as follows:

where 𝑥𝑖𝑗𝑘 is the value of the 𝑖𝑡ℎ feature map at position 𝑗, 𝑘; 𝑝 is the vertical index in the local region; 𝑞 is the horizontal indicator in the local region; and 𝑦 is the consequence of this process in the pooling layer. A convolutional layer with more significant strides might be used to replace the pooling layer, resulting in a reduction in spatial size.

### 3.5.3 Fully connected layer

Any number of ultimately linked layers may be added after the previous ones. These are typical neural networks composed of neurons linked to all activations in the previous layer.

### 3.5.4 Backpropagation in deep convolutional neural network

Convolutional and pooling procedures are applied to the gradient in a DCNN model as part of the backpropagation method. The gradient is calculated as follows for the convolutional operation:

where J is the cost function, (W, b) are the parameters, and (x, y) is the training information and label sets (Ufldl.stanford.edu., 2021).Similarly, the gradient for the pooling layer is determined as follows:

K denotes the filter’s index, and h′(aj) denotes the activation function’s derivative. Additionally, the up-sample process generates the error throughout the pooling layer by verifying the error for each neuron that enters this layer (Ufldl.stanford.edu., 2021).

### 3.5.5 Softmax classifier

In complicated situations, binary classification is inadequate since most real-world information is classified into numerous classifications. Multi-class classification is necessary in certain instances. This can be accomplished through the use of a classifier such as softmax. In contrast to additional classifiers, which output unadjusted results for each class, the softmax classifier outputs K mutually exceptional normalised class probabilities (Goodfellow et al., 2016). The outputs of this classifier are construed as 𝑦𝑘 (𝑥, 𝑤) = 𝑝(𝑡𝑘 = 1|𝑥) with the following error function::

The softmax classifier reduces a K-dimensional vector z of arbitrary real-valued scores to a vector of values between 0 and 1 that total 1. This is accomplished through the use of the softmax function (Goodfellow et al., 2016):

which satisfies 0 ≤ 𝑦𝑘 ≤ 1 and ∑𝑘 𝑦𝑘 ≤ 1

### 3.5.5 Discrete convolution

The discrete convolution process is a critical component of the CNN since it is utilised to calculate the values before the activation. These amounts before the activation are then subjected to a non-linearity to obtain the values of the feature maps’ hidden units (Shocher et al., 2020). The following formula expresses the discrete convolution of an input picture x with an (r \*r)-kernel k:

A screenshot of a computer

Description automatically generated with low confidencewhere (𝑥 ∗ 𝑘)𝑖𝑗 is the convolution at position (𝑖, 𝑗), 𝑥𝑖+𝑝,𝑗+𝑞 is the value of the input image at position (𝑖 + 𝑝, 𝑗 + 𝑞), and 𝑘𝑟−𝑝,𝑗−𝑞 is the value of the kernel at position (𝑟 − 𝑝, 𝑟 − 𝑞). Figure 9 illustrates a discrete convolution operation:

Figure 9.Example of discrete convolution.

The hidden units’ values are calculated as follows:

where 𝑘𝑖𝑗 is the convolution kernel, 𝑥𝑖 is the 𝑖 𝑡ℎ channel of input, and 𝑔(⋅) is the ReLU activation function. It is worth mentioning that due to the non-linearity of the discrete convolution process, the hidden layers can detect features by stressing the relationship between a learned filter and a particular region. This combination aids in the detection of edges and wrinkles (Shocher et al., 2020).

### 3.5.6 Deep convolutional neural network architectures

When the layers outlined in Section 3.1 are combined, they result in DCNN designs. DCNNs’ most typical implementation stacks many convolutional and pooling layers collectively until the input picture is spatially diminished. Following this transitional output, there are fully linked layers, the last of which outputs a value such as the class probability (Goodfellow et al., 2016; Cs231n.github.io., 2021). While DCNNs are robust neural networks, it is critical to keep the following factors in mind while designing them::

#### 3.5.6.1 Input layer

The input volume should be divisible by two. These sizes range between 32 and 512. (Goodfellow et al.,2016),( Cs231n.github.io. ,2021)

#### 3.5.6.2 Convolutional layer

Tiny filters with a stride of one and no cushioning are often employed. This zero padding is chosen using the principle P = F - 1 / 2 to maintain the magnitude of the input volume. It is notable that although tiny filters like 3x3 and 5x5 may be applied in any layer, larger filters like 7x7 must be reserved for the first convolutional layer (Goodfellow et al., 2016; Cs231n.github.io., 2021). It is better to stack many tiny filters than to use a single large filter equivalent because the small filters preserve the non-linearities of the input and need fewer parameters (Goodfellow et al., 2016; Cs231n.github.io., 2021).

#### 3.5.6.3 Pooling layer

2x2 or 3x3 filters are often used. The rationale for preferring tiny filters to large filters is because large filters are very lossy and aggressive, resulting in poor performance (Goodfellow et al., 2016; Cs231n.github.io., 2021).

#### 3.5.6.4 Strides and zero-padding

Goodfellow et al. (2016) and Cs231n.github.io. (2021) stated that a stride of one is preferable since it maintains the spatial dimensions of the input volume and is more practical. Similarly, zero padding preserves the input’s spatial dimensions and prevents the information near the border from dissipating too fast.

### 3.5.7 Strategies for enhancement

#### 3.5.7.1 Network reduction

As is well known, noise learning is a significant contributor to overfitting. As a result, noise reduction becomes a study focus for overfitting inhibition. Pruning is suggested as a way to decrease the size of final classifiers in relational learning, particularly decision tree learning. Pruning is a significant idea used to decrease classification complexity by removing unimportant or unnecessary data, avoiding overfitting, and enhancing classification accuracy..

#### 3.5.7.2 Regularisation

Generally, many characteristics may have an effect on the output of a model. The model becomes more complex as the number of features increases. An overfitting model often includes all characteristics, even if some have a negligible impact on the final output. Alternatively, even worse, some of these characteristics are meaningless sounds for the output. To minimise these instances, two options can be used: remove any superfluous characteristics from the model and reduce the weights of elements that have little effect on the final categorisation. To do so, a penalty term is applied, which is called regularisation. There are three types of regularisation: L1 regularisation, L2 regularisation, and dropout (Rao et al., 2018; Ying, 2019).

#### 3.5.7.3 Early stopping

Early stopping is a technique that defines an arbitrarily large number of training epochs and then terminates training when the model’s performance on the validation dataset stops increasing (Prechelt, 2012; Rao et al., 2018; Ying, 2019).

#### 3.5.7.4 Batch normalisation

Batch normalisation is a method for training very deep neural networks in which the inputs to a layer are standardised for each mini-batch. This results in a stabilisation of the learning process and a significant decrease in the number of training epochs needed to train deep networks (Ioffe & Szegedy, 2015).

#### 3.5.7.5 Keras Tuner

The Keras Tuner is a package that assists researchers in selecting the optimum set of hyper-parameters for their application. Hyper-parameter tuning, or hyper-tuning, is the process of choosing the optimal collection of hyper-parameters for one’s machine learning application (Tensorflow, 2021).

#### 3.5.7.6 ModelCheckpoint

The ModelCheckpoint callback class makes it possible to define where to checkpoint the model weights, how the file should be named, and under what circumstances to make a checkpoint of the model (Tensorflow, 2021).

#### 3.5.7.7 Data augmentation

The algorithm is not the sole factor influencing the final classification accuracy in machine learning. In many instances, particularly in the field of SL, its performance may be substantially influenced by the amount and quality of training data (Ying, 2019).

A close-up of a logo

Description automatically generated with low confidenceData augmentation is a process of harnessing more images from the ones that are already used. This method applies different modifications such as rotation, scaling, zooming, changing the brightness, or flipping the picture (Shorten & Khoshgoftaar, 2019).

Figure 10.Data augmentation procedure

# Synthesis

# Chapter 4. Project research

This section explains the approach utilised to create the project and details each step. In addition, it describes the components used to create the method, the rationale for their selection, and the dataset used to train the neural networks.

## 4.1 AffectNet dataset

Automated emotional computing ITW is a difficult problem in computer vision. There are few annotated datasets of facial expressions ITW that include more than single emotions. Within the continuous dimensional display, there are very restricted remarks on face databases for personal computing. This project makes use of AffectNet data. Because the database only includes facial pictures, no other kind of data is gathered. This database contains the largest collection of naturalistic facial expressions, valence, and arousal, allowing the study of automated facial expression identification using two distinct emotion models. The database was developed via the collection and annotation of facial pictures. The term “affect” is a psychological concept that refers to the outward expression of feelings and emotions (Mollahosseini et al., 2019).

The AffectNet database contains approximately 1 million facial pictures gathered from the internet by questioning three significant web search engines utilising 1,250 emotion-related watchwords in six different languages. Approximately 550,000 images were automatically annotated using the ResNext neural network. These samples were obtained using the other samples as a training set. The other 420,000 samples are manually annotated for the presence of seven discrete facial expressions and the intensity of valence and arousal (Mollahosseini et al., 2019). Table 1 provides a summary of the different datasets used for facial emotion recognition made by Mollahosseini et al. (2019).

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Dataset** | **Database information** | **Number of samples** | **Condition** | **Affect Modeling** |
| CK+ | * Frontal and 30 degree images | 123 | * Controlled * Posed | * 30 AUs * seven emotion categories |
| MultiPie | * Approximately 750,000 images * Under multiple viewpoints and illuminations | 337 | * Controlled * Posed | * seven emotion categories |
| MMI | * Subjects portrayed 79 series of facial expressions * Image sequence of frontal and side view are captured | 25 | * Controlled * Posed * Spontaneous | * 31 AUs * Six basic expression |
| DISFA | * Video of subjects while watching a four minutes video * clip are recorded by a stereo camera | 27 | * Controlled * Spontaneous | * 12 AUs |
| SALDB | * SAL * Audiovisual (facial expression,shoulder, audiocues) * 20 facial feature points, five shoulder points for video | 4 | * Controlled * Spontaneous | * Valence Quantized * Continuous |
| RELOCA | * Multi-modal audio, video, ECG and EDA | 46 | * Controlled * Spontaneous | * Valence and arousal (continuous) * Self assessment |
| AM-FED | * 242 facial videos | 242 | * Spontaneous | * 14 AUs |
| DEAP | * 40 one-minute long videos shown to subjects * EEG signals recorded | 32 | * Controlled * Spontaneous | * Valence and arousal * Self assessment |
| AFEW | * Videos | 330 | * Wild | * seven emotion categories |
| FER-2013 | * Images queried from web | Approximately 35000 | * Wild | * seven emotion categories |
| EmotioNet | * Images queried from web * 100,000 images annotated manually * 900,000 images annotated automatically | Approximative 100000 | * Wild | * 12 AUs annotated * 23 emotion categories based on AUs |
| Aff-Wild | * 500 videos from YouTube | 500 | * Wild | * Valence and arousa |
| FER-Wild | * 24,000 images from web | Approximately 24000 | * Wild | * seven emotion categories |
| AffectNet | * 1 million images with facial landmarks * 450,000 images annotated manually | Approximately 450000 | * Wild | * eight emotion categories * Valence and arousal |

Table 1.Facial emotion recognition datasets

The AffectNet dataset provides 11 annotated emotions: neutral, happiness, sadness, surprise, fear, disgust, anger, contempt, none, uncertain, and no face.

|  |  |
| --- | --- |
| Neutral | 75,374 |
| Happy | 134,915 |
| Sad | 25,959 |
| Surprise | 14,590 |
| Fear | 6,878 |
| Disgust | 4,303 |
| Anger | 25,382 |
| Contempt | 4,250 |
| None | 33,588 |
| Uncertain | 12,145 |
| Non-Face | 82,915 |
| Total | 420,299 |

Table 2.Number of samples in each class

Features provided by the AffectNet database:

* Images of the faces
* Location of the faces in the images
* Location of the 68 facial landmarks
* Eleven emotion and non-emotion categorical labels (Neutral, Happiness, Sadness, Surprise, Fear, Disgust, Anger, Contempt, None, Uncertain, No-Face)
* Valence and arousal values of the facial expressions in a continuous domain

## 4.2 Transfer learning

Transfer learning is a machine learning method in which an existing model produced for one job is utilised as the basis for another task. The benefit of this method is that it substantially lowers the amount of time and money required for computational calculation (Torrey & Shavlik, 2010; Weiss et al., 2016; Lu et al., 2015). Transfer learning for image classification is based on the premise that an effective generic model of the visual world may be produced by training a model on a sufficiently large and diverse dataset. It is then possible to use these learned feature maps to train a large model on a large dataset without starting from scratch (Torrey & Shavlik, 2010; Weiss et al., 2016; Lu et al., 2015). The only change required is to unfreeze a few of the top layers of a frozen model base and train the newly added classifier levels and the base model’s final layers concurrently. This enables the higher-order feature representations to be fine tuned to be more relevant to this job (Torrey & Shavlik, 2010; Weiss et al., 2016; Lu et al., 2015).

## 4.3 Pre-trained modes

There are multiple pre-trained deep neural networks available to be fine tuned. Among these networks, the ones used to develop the facial emotion recognition algorithm are Resnet101V2, VGG19, InceptionV3, and MobileNetworkV2.These networks were trained on the ImageNet datasets, which contained over 1 million pictures divided into 1,000 classes (ImageNet, 2021).

## 4.3.1 ResNet101V2 model

This model was created by He et al. (2015). Only colour images were used for the training part, which were sized to 224\*224 pixels. As the name suggests, the model was created by stacking 101 residual layers. The convolutional layers have mostly 3\*3 filters. The number of neurons in each layer remains the same until the feature map is halved. Once the feature map is halved, the number of neurons is doubled. The network ends with a global average pooling layer and a fully connected layer of 1,000 neurons (He et al., 2015).

The model’s architecture is outlined in Table 3.

|  |  |  |
| --- | --- | --- |
| Layer | Output size | Architecture |
| Convolutional | 112\*112 | 7\*7 ,64, stride 2 |
|  |  | 3\*3 max pooling, stride 2 |
| Convolutional | 56\*56 | |  |  | | --- | --- | | 1\*1, 64 |  | | 3\*3, 64 | \*3 | | 1\*1,256 |  | |
| Convolutional | 28\*28 | |  |  | | --- | --- | | 1\*1, 128 |  | | 3\*3, 128 | \*8 | | 1\*1,512 |  | |
| Convolutional | 14\*14 | |  |  | | --- | --- | | 1\*1, 256 |  | | 3\*3, 256 | \*36 | | 1\*1,1024 |  | |
| Convolutional | 7\*7 | |  |  | | --- | --- | | 1\*1, 512 |  | | 3\*3, 512 | \*3 | | 1\*1,2048 |  | |
|  |  | Global average pool |
| Dense | 1\*1 | 1000-d fc, softmax |

Table 3.ResNet101V2 architecture

### 4.3.2 VGG19 model

T The model proposed by Simonyan and Zisserman (2015) uses the same input size of 224\*224 RGB pictures. This model is not as deep as ResNet101V2, but the complexity is much higher. This model has a total of 16 convolutional layers and three fully connected layers. This model uses kernels with a size of 3\*3 with a stride size of 1 pixel. The chosen pooling method is max pooling with a window of 2\*2 and a stride of 2. Thus, the architecture ends with three fully connected layers. Two of these layers have 4,096 neurons, and the last one has an output of 1,000. This architecture is described in Table 4.

|  |  |  |
| --- | --- | --- |
| Layer | Output size | Architecture |
| Convolutional | 224\*224 | (3\*3, 64 )\*2 |
|  |  | 3\*3 max pooling, stride 2 |
| Convolutional | 112\*112 | (3\*3, 128 )\*2 |
|  |  | 3\*3 max pooling, stride 2 |
| Convolutional | 56\*56 | (3\*3, 256 )\*4 |
|  |  | 3\*3 max pooling, stride 2 |
| Convolutional | 28\*28 | (3\*3, 512)\*4 |
|  |  | 3\*3 max pooling, stride 2 |
| Convolutional | 14\*14 | (3\*3, 512)\*4 |
|  |  | 3\*3 max pooling, stride 2 |
| Dense | 7\*7 | (1\*1,4096) \*2 |
| Dense | 1\*1 | 1000-d fc, softmax |

Table 4.VGG19 architecture

### 4.2.3 InceptionV3 model

InceptionV3 has a unique architecture, the rest on the CNN mentioned before. The input data uses 299\*299 colour images. Moreover, the developers chose to create a model using inception blocks, which is a different strategy that was initially presented in GoogleNet (Szegedy et al., 2015). Compared to other networks, an inception block allows multiple operations to be executed in parallel. Moreover, it utilises multiple convolutional sizes in one block, such as 1 × 1, 3 × 3, and 5 × 5 (Szegedy et al., 2015). The Inception block is represented in Figure 11.

Diagram

Description automatically generated

Figure 11.Representation of an Inception block

The whole architecture of the model contains more than one inception block as shown in Table 5.

|  |  |  |
| --- | --- | --- |
| Layer | Output size | Architecture |
| Convolutional | 149\*149 | (3\*3, 32)\*2 |
|  |  | 3\*3 max pooling, stride 2 |
| Convolutional | 73\*73 | (3\*3, 64) |
| Convolutional | 71\*71 | (3\*3, 80 ),stride 2 |
| Convolutional | 35\*35 | (3\*3, 192 ) |
| Inception block | 35\*35 | (288) \*3 |
| Inception block | 17\*17 | (768) \*5 |
| Inception block | 8\*8 | (1280) \*3 |
|  |  | 8\*8 max pooling |
| Dense | 1\*1 | (2048) |
| Dense | 1\*1 | 1000-d fc, softmax |

Table 5.InceptionV3 architecture

### 4.3.4 MobileNetV2 model

Diagram, table

Description automatically generated Howard et al. (2017) created this model to be more efficient than other networks. Additionally, it has an input size of 224\*224 and uses RGB images. For this architecture, Howard et al. created a depthwise separable convolutional layer. Separable depthwise convolutions are composed of two layers: depthwise and pointwise convolutions. This factorisation results in a significant reduction in calculation time and model size.

Figure 12.Representation of depthwise and pointwise convolutions

The creators apply a single filter to each input channel through depthwise convolutions (input depth). The output of the depthwise layer is then combined linearly using pointwise convolution (a basic 1\*1 convolution; Howard et al., 2017).

|  |  |  |
| --- | --- | --- |
| Layer | Output size | Architecture |
| Convolutional | 224\*224 | (3\*3, 32 ),stride 2 |
| Convolutional depthwise | 112\*112 | (3\*3, 32 ) |
| Convolutional | 112\*112 | (1\*1, 64 ) |
| Convolutional depthwise | 112\*112 | (3\*3, 64 ),stride 2 |
| Convolutional | 56\*56 | (1\*1, 128 ) |
| Convolutional depthwise | 56\*56 | (3\*3, 128) |
| Convolutional | 56\*56 | (1\*1, 128) |
| Convolutional depthwise | 56\*56 | (3\*3, 128 ),stride 2 |
| Convolutional | 28\*28 | (1\*1,256) |
| Convolutional depthwise | 28\*28 | (3\*3,256) |
| Convolutional | 28\*28 | (1\*1,256) |
| Convolutional depthwise | 28\*28 | (3\*3,256), stride 2 |
| Convolutional | 14\*14 | (1\*1,512) |
| 5\*Convolutional depthwise  5\*Convolutional | 14\*14  14\*14 | (3\*3,512)  (1\*1,512) |
| Convolutional depthwise | 14\*14 | (3\*3,512), stride 2 |
| Convolutional | 7\*7 | (1\*1,1024) |
| Convolutional depthwise | 7\*7 | (3\*3,1024), stride2 |
| Convolutional | 7\*7 | (1\*1,1024) |
|  |  | 7\*7 average pooling |
| Dense | 1\*1 | 1000-d fc, softmax |

Table 6.MobileNetV2 architecture

## 4.4 Hardware resources

Only the personal PC was used in the first stages of development, but the computational time required was too much, so alongside the PC, the researcher chose to use Google Colaboratory Pro, which is an online service platform.

### 4.4.1 Personal Computer

To complete such a demanding task, the computer requires a significant amount of computing power. Three essential components are required for model development: RAM, a central processing unit (CPU), and a graphics processing unit (GPU). When just a small quantity of data is used to build a model, the GPU may not be required since the CPU performs the computation. Typically, large datasets are needed to build a decent model, and as a result, it may take days, weeks, or months to develop a good model using just the CPU. In this instance, the GPU was utilised to speed the learning process (NVIDIA, 2021). For this project, an Intel Core i7 – 10,750 CPU was used, and there was 32G of RAM memory available for this project. The GPU used was an NVIDIA GeForce RTX-2070 with a memory of 8 GB. Using all the resources at almost 100% capacity, the computer could perform the training process required.

### 4.4.2 Google Colaboratory web service

Colaboratory (Colab) is a Google Research product. Colab enables anybody to create and run arbitrary Python code directly from the browser, making it ideal for machine learning, data analysis, and teaching. Colab is a hosted Jupyter Notebook service that needs no configuration and provides access to computational resources such as GPUs (Google, 2021). Virtual computers power notebooks with a maximum lifespan of up to 12 hours. Notebooks also disconnect from VMs if they are kept inactive for an extended period. Maximum virtual machine lifespan and idle timeout may change over time or according to the total use of resources

The GPUs that are available in Colab change over time. This is essential for Colab to continue providing free access to these materials. Nvidia K80s, T4s, P4s, and P100s are often accessible as GPUs in Colab. There is no method in Colab to choose the kind of GPU to which one may connect at any given moment. Colab Pro may be of interest to those seeking more reliable access to Colab’s fastest GPUs (Google, 2021).

## 4.5 Programming language

Selecting the appropriate programming language for the system’s development was difficult since there are currently over 2,000 high-level programming languages. Fortunately, the project requirements eased this issue. During the training phase, the programming language needed to be easy to learn and use, and it needed to be compatible with the GPU libraries since the project included neural networks. Additionally, sufficient documentation and community assistance were required to ensure that issues encountered throughout the development cycle were resolved quickly.

After taking these variables into account, the list was significantly condensed. Python, Matlab, R, and C++ are among the most used remaining programming languages for machine learning (Brownlee, 2014; Aruoba & Fernández, 2015). Python is a user-friendly programming language that has a range of libraries that enable transparent GPU usage (Brownlee, 2014; Aruoba & Fernández, 2015). Matlab is a balanced option that is user friendly and quick and has adequate performance but requires a licence to utilise the main product and certain of its libraries (Brownlee, 2014; Aruoba & Fernández, 2015). R is a strong language, but its learning curve is high (Brownlee, 2014; Aruoba & Fernández, 2015). C++ is considered the best choice when speed is essential, but it is difficult to learn and more difficult to optimise. Additionally, the development cycle is much longer than that of other programming languages (Brownlee, 2014; Aruoba & Fernández, 2015). Following a thorough study, Python was chosen to train the CNNs because of its high performance, easy learning curve, and lack of licencing requirements.

## 4.6 Software and libraries

* **Anaconda** is a Python distribution that comes with over 400 scientific, mathematical, engineering, and data analytic programmes. It is simple to install and enables interaction with different languages (Anaconda, 2021).
* **Jupyter Notebook** is a free, open-source, interactive online application referred to as a computational notebook that enables academics to integrate software code, computational output, explanatory text, and multimedia resources into a single document.
* **CUDA** is a parallel computing platform and programming paradigm developed by NVIDIA that enables the GPU’s capabilities to be fully used.
* **CuDNN** is an acronym for CUDA Deep Neural Network library, a collection of primitives for DNN that implement common functions such as forward and backward convolution, pooling, normalisation, and activation layers in a highly optimised manner (NVIDIA, 2021).
* **Tensorflow** is a framework that ensures access to powerful APIs such as Keras from which different pre-trained models and functions can be imported to build a robust system. Additionally, it can run on both GPUs and CPUs, which decrease the computational time (Tensorflow, 2021).
* **Keras** is an API that adheres to best practices for cognitive load reduction: it provides consistent and simple APIs, reduces the number of user activities needed for typical use cases, and delivers clear and actionable error signals. Additionally, it includes comprehensive documentation and development instructions (Keras, 2021).
* **matplotlib.pyplot** is a set of methods that emulate the functionality of MATLAB in matplotlib. Each pyplot function modifies a figure in some way; for example, it produces a figure and a plotting region inside the figure, plots some lines within the plotting area, and decorates the plot with labels (Matplotlib, 2021).
* **NumPy is** the foundational Python module for scientific computing. It is a Python library that includes a multidimensional array object, various derived objects (such as masked arrays and matrices), and a collection of routines for performing fast array operations, such as mathematical, logical, shape manipulation, sorting, selecting, I/O, discrete Fourier transforms, introductory linear algebra, basic statistical operations, and random simulation (NumPy, 2021).
* **OS** enables the use of operating system-dependent functionality in a portable manner. It enables the user to read and write files, modify paths, and view the contents of all files on the command line. Additionally, it can create temporary files and directories and manage high-level files and directories (Docs.python.org., 2021).
* **Scikit-**learn is arguably the most helpful machine learning package available in Python. The sklearn library provides a variety of fast machine learning and statistical modelling techniques, such as classification, regression, clustering, and dimensionality reduction (Scikit-learn.org., 2021).
* **Pandas** is based on two essential Python libraries: matplotlib for data visualisation and NumPy for mathematical calculations. Pandas serves as a wrapper over these libraries, enabling the user to access numerous matplotlib and NumPy functions quickly. Before pandas, most analysts utilised Python for data munging and preparation before transitioning to a more domain-specific language like R for the remainder of their workflow (Pandas, 2021).
* **Itertools** provides a collection of iterator building pieces inspired by APL, Haskell, and SML techniques. Each has been rewritten in Python-friendly code. The module standardises a foundational collection of quick, memory-efficient utilities that may be used alone or in combination. They combine to create an “iterator algebra” which enables the rapid and efficient construction of customised tools in pure Python.

## 4.4 Tuning the hyper-parameters

Accurate model training requires a critical step known as hyper-parameter tweaking. This procedure focuses on fine-tuning the hyper-parameters, which are the model’s values, to obtain optimal performance. The following hyper-parameters were adjusted during this phase:

**Learning rate:** This limits the number of iterations required for the model to produce an acceptable solution by adjusting the step size when approaching the global or local minima. This is often the single most critical hyper-parameter, and it should be constantly adjusted. The values of a neural network may be very varied, ranging from more than 106 to fewer than 1. However, for ordinary neural networks, a default value of 0.01 is sufficient (Cs231n.github.io., 2021; Prechelt, 2012).

**Gradient descent optimisation algorithm**: This hyper-parameter influences convergence time since it defines the gradient descent method to employ. Six algorithms are often used: accelerated gradient, momentum, Adagrad, Adadelta, RMSprop, and Adam.

**Number of epochs**: This determines the duration of the training process. On the one hand, a small number prevents convergence by prematurely terminating the training process; on the other hand, a large number wastes processing time without improving performance. It is critical to note that when early stopping is used, there is no need to specify the number of epochs; the model will terminate as soon as performance improves or declines (Prechelt, 2012; Hastie, 2001).

**Number of dense layers:** This defines the number of layers, which influences the kind of decision boundary the design may represent. However, as the number of layers increases, the model becomes more complicated, and it becomes more difficult to choose appropriate parameters that allow the architecture to converge.

**Convolutional filter size:** This parameter determines how the DCNN responds to certain types of features. When the filter is large, such as 7x7 or more, the network may miss critical information in the input. When the filter is tiny, the network considers more information, which is not necessarily accurate. Convolutional filters of sizes 3x3 and 5x5 are suggested for all layers; however, 7x7 filters should be used only in the initial layer.

**Number of convolutional filters:** This parameter determines the number of activation maps and the characteristics to which the network responds.

**Activation function**: This regulates neuronal activity. When the design is composed of many layers, this is a critical hyper-parameter. As previously stated, disappearing and exploding gradients are two backpropagation effects intimately related to the starting weights and activation function. Cs231n.github.io. (2021) asserts that the ReLU and maxout activation functions are preferable because they address these issues.

# Chapter 5. Experiments and results

This chapter describes all the stages and approaches taken to develop a robust model. This includes the modifications made step by step and the fine-tuning process. In order to start the training, the dataset had to be transferred into the system. Only the manually annotated pictures were used in these experiments, with 283,901 images matching the following emotions: anger, disgust, fear, happiness, neutral, sadness, and surprise. Because of the size of the dataset, the pictures could not be fed to the algorithm as a whole. Due to this problem, mini-batch sizes had to be used. . ImageDataGenerator() class, along with .flow\_from\_directory(), allow the use of these mini-batch sizes and provide access to use data augmentation (Tensorflow, 2021).

As previously mentioned, this study used four pre-trained models that were fine tuned. The models are VGG-19, InceptionV3, ResNet101V3, and MobileNetV3. The input shape chosen for this model is 224\*224 since this is the value previously used to train the models on the ImageNet dataset. The top layers were not included in order to be able to fine-tune the model. After some calculation, the batch size used for this project was 128, which is a value that fits the GPU and RAM requirements. Data augmentation was used on the dataset to increase the generalisation capabilities of the models. This includes data normalisation, a 0 to 10% zooming range, a 5-degree rotation range, and horizontal flip. Here, the dataset was divided into pictures for training (80%) and validation (20%). Furthermore, the pictures for training were shuffled to increase the generalisation capabilities..

## 5.1. First configuration models

For the first experiment, the first layer added was a global max pooling added to reduce the number of parameters as Kim and Jeong (2019) suggested. Before the data is fed to the fully contacted layers, it must go through a flattened layer. This converts the data received from the convolutional layers into a one-dimensional array (Jeong, 2019). A fully connected layer with 1,024 neural units was added using ReLu as an activation function. This activation function was used because it utilises half as many parameters as the maxout (Cs231n.github.io., 2021). The output layer has only seven neurons, matching the number of emotions with a softmax activation function. Finally, the model was compiled using the Adam optimiser since this optimiser is the most efficient of them all (Kingma & Ba, 2015). The model trained for 10 epochs, and the best weights were saved using the ModelCheckpoint tool. This ensured that the model created achieved the best result using the current setup.

Figure 13.Training results for each model in first stage

Application

Description automatically generatedGraphical user interface, application

Description automatically generatedGraphical user interface, application, email

Description automatically generatedVgg19
 As shown in Figure 13, Inception, VGG-19, and MobileNet all performed similarly. In contrast, ResNet had a higher generalisation rate of 5 to 7%. At this point, none of the models showed indications of overfitting or underfitting. This is shown by the identical values of validation accuracy and accuracy or validation loss and loss.

Figure 14.VGG19-first stage

Figure 15.ResNet101V2-first stage

Table

Description automatically generated with medium confidenceChart, table

Description automatically generated with medium confidenceChart

Description automatically generatedChart

Description automatically generated In the figures above, diagrams show the accuracy and cross-entropy for each model train. The cross-entropy of a random variable or collection of events measures the difference between the probability distributions. This information is the number of bits needed to encode and convey an event. Occurrences with a low probability include more information, whereas events with a higher probability include less information. At this point, it is evident that the training process for each model is stable due to the low fluctuation between training accuracy and validation accuracy and between training loss and validation loss. The confusion matrix for each model is shown in this figures:

Figure 16.InceptionV3-Confusion matrix-first stage

Figure 17.MobileNetV2-Confusion matrix-first stage

Figure 18.VGG19-Confusion matrix-first stage

Figure 19.ResNet101V2-Confusion matrix-first stage

Figure 20.InceptionV3-first stage

Figure 21.MobileNetV2-first stage

The classification of the 56,777 validation images into their respective classes can be seen above. At first glance, the pictures demonstrate that each model has an excellent ability to distinguish between happy and neutral images. Additionally, each model seems to incorrectly categorise other classes as happy or neutral. The disgust class had the lowest accuracy rate of any class, with each model achieving less than 0% accuracy. The ResNet101V2, which has the highest overall accuracy, produced the pattern that creates the matrix’s more noticeable diagonal. Additionally, the ResNet101V2 is the only model capable of classifying fear images with an accuracy greater than 0% with a 16% classification rate.

Table

Description automatically generatedTable

Description automatically generatedTable

Description automatically generatedTable

Description automatically generatedTo understand the classification report, the definitions of precision, recall, and F1-score must be understood. The precision of a prediction is defined as the ratio of correctly predicted positive observations to all anticipated positive observations. The recall ratio is the number of adequately predicted positive observations divided by the total number of observations in the actual class. The F1 score is calculated as the weighted average of precision and recall. As a result, this score accounts for both false positives and negatives. While F1 is not as intuitive as accuracy, it is often more helpful than accuracy, mainly when the class distribution is unequal.

Figure 22.InceptionV3-Clasification report-first stage

Figure 23.MobileNetV2-Clasification report-first stage

Figure 24.ResNet101V2-Clasification report-first stage

Figure 25.VGG19-Clasification report-first stage

Knowing the meeting of the metrics specified above, it is possible to discuss the performance of each model in every class. In order to obtain good accuracy, the F1-score needs to be high. In the diagram above, all the models struggle on recall for disgust, fear, and surprise. However, the ResNet101V2 surpasses all the models in those categories, scoring the highest recall and F1-score. As for VGG19, the recall, which tends towards 0, makes it only archive 0.1 at F1-score even if it has perfect precision.

## 5.2. Second configuration models

The next step consisted of changing the dense layer with 1,024 neuros into two layers: one of 512 and one of 256. This experiment shows whether two layers with fewer parameters can perform more efficiently and be more accurate.

Figure 26.Training results for each model in second stage

Using these two layers did not produce a significant increase in accuracy, but the model developed was compiled more quickly due to the smaller number of parameters.

Graphical user interface, application

Description automatically generated with medium confidenceGraphical user interface, application

Description automatically generatedApplication

Description automatically generated with medium confidenceGraphical user interface, application, table

Description automatically generatedFor both stages enunciated above, more experiments were conducted to find the best set of parameters. The number of neurons used was between 32 and 2,048. All the numbers chosen were powers of two. Therefore, the two charts above present only the best results achieved.

Figure 27.InceptionV3-seond stage

Figure 28.MobileNetV2-second stage

Figure 29.VGG19-second stage

Figure 30.ResNet101V2-second stage

Table

Description automatically generatedChart

Description automatically generated with low confidenceChart

Description automatically generated with medium confidenceGraphical user interface

Description automatically generated with low confidenceChanging the parameters did not make a substantial change in the learning process. As the diagrams show, the models’ accuracy increased smoothly as before. However, the ResNet101V2 made a little exception at the beginning of the training and towards the end when it underfitted. This is shown in the diagram as the blue line going higher than the orange one, but it came back to normal in the end.

Figure 31.InceptionV3-Confusion matrix-second stage

Figure 32.MobileNetV2-Confusion matrix-second stage

Figure 33.ResNet101V2-Confusion matrix-second stage

Figure 34.VGG19-Confusion matrix-second stage

Table

Description automatically generatedTable

Description automatically generatedTable

Description automatically generatedTable

Description automatically generatedDecreasing the number of parameters did not substantially affect VGG19 and InceptionV3 in terms of the error rate. The MobileNetV2 suffered an error increase of 8% since the accuracy dropped from 26 to 18%. The 8% was misclassified as neutral and sad. On the other hand, the ResNet101V2 increased the angry class’s accuracy by 6% while the other remained unchanged. Unfortunately, none of the models were able to decent accuracy for the disgust class at this point.

Figure 35.ResNet101V2-Clasification report-second stage

Figure 36.VGG19-Clasification report-second stage

According to the categorisation report, this technique was advantageous for MobileNetV2 and VGG19 when it came to the anger class. This is because the VGG19 had a greater precision but a lower recall, while the MobileNetV2 had a higher recall. Regrettably, the F1 score for InceptionV3, MobileNetV2, and VGG19 was 0. As a result, it is impossible to conclude that no additional major changes occurred.

Figure 37.Inceptionv3-Clasification report-second stage

Figure 38.MobileNetV2-Claification report-second stage

## 5.3. Third configuration models

Figure 39.Optimizer comparison for each model

The experiments previously conducted only used the Adam for the optimisation of the models. Here each of the models were tested using different optimisers like Adadelta and Adagrad. This experiment was conducted to demonstrate the theory that Adam has all the benefits of Adadelta and Adagrad without the drawbacks, as Kingma and Ba (2015) have mentioned.

Figure 40.Optimization comparison

Graphical user interface, application

Description automatically generatedGraphical user interface, text, application

Description automatically generated As it can be seen in the Adam scores the best validation accuracy for each model. This test demonstrated that experiments conducted by Kingma and Ba (2015) are applicable in the case of each model. Even if in this case Adam performed better, this does not mean that Adagrad and Adadelta should not be used. There might be problems in which they could outperform the other optimisers.

Figure 41.ResNet101V2-Adagrad

Figure 42.VGG19-Adagrad

Table

Description automatically generatedTable

Description automatically generated with medium confidenceTable

Description automatically generated with medium confidenceChart

Description automatically generated with medium confidenceGraphical user interface, application

Description automatically generatedGraphical user interface, application

Description automatically generated Even the Adagrad, which exhibited no indications of overfitting or underfitting, performed well with the two completely linked layers of 512 and 256. All of the models follow the same pattern as shown by the orange and blue lines. Again, the ResNet101V2 exhibits the highest sloop in the same number of epochs as the others.

Figure 43.InceptionV3-Confusion matrix-Adagrad

Figure 44.MobileNetV2-Confusion matrix-Adagrad

Figure 45.ResNet101V2-Confusion matrix-Adagrad

Figure 46.VGG19-Confusion matrix-Adagrad

Figure 47.InceptionV3-Adagrad

Figure 48.MobileNetV2-Adagrad

Table

Description automatically generatedTable

Description automatically generatedEven if the validation accuracy is similar to that of Adam, the confusion matrix reveals that specific models, such as VGG19 and MobileNetV2, have a more significant error for all classes except happy and neutral. This can be observed in the matrix, where the boxes corresponding to the happy and neutral classes become darker. When it comes to ResNet101V2 and InceptionV3, there is discernible difference in the error rate for any class. InceptionV3 and VGG19 had a precision of 0%, whereas ResNet101V2 and MobileNetV2 had a precision of 1 to 3% .

Figure 49.ResNet101V2-Clasification report-Adagrad

Figure 50.VGG19-Clasification report-Adagrad

A picture containing text, indoor

Description automatically generatedTable

Description automatically generated

Figure 51.InceptionV3-Clasificaion report-Adagrad

Figure 52.MobileNet2-Clasification report-Adagrad

Line chart

Description automatically generatedGraphical user interface, application

Description automatically generatedGraphical user interface, application

Description automatically generated with medium confidenceGraphical user interface, application

Description automatically generated The categorisation report indicates a substantial reduction in accuracy and recall for each model, most notably for the anger class. Additionally, InceptionV3, VGG19, and MobileNetV2 show that disgust and the f1-score both have a recall of 0%. Fear recall decreased significantly to less than 5% for MobileNetV2 and IneptionV3 and 0% for the VGG19. The findings show that this optimiser prevents models from achieving a higher overall generalisation rate and has significant difficulties identifying anger, disgust, and fear.

Figure 53.RestNet101V2-Adadelta

Figure 54.VGG19-Adadelta

Figure 55.MobileNetV2-Adadelta

Figure 56.InceptionV3-Adadelta

The Adadelta optimiser produced a similar learning pattern as Adam and Adagrad even if it had the lowest accuracy. It did not show any sign of overfitting or underfitting. Comparing the slopes of each model indicates that even if the accuracy increases at a slow rate, they would learn more if the number of epochs increased.

Table

Description automatically generatedA picture containing text, indoor

Description automatically generatedTable

Description automatically generatedChart

Description automatically generatedTable

Description automatically generatedTable

Description automatically generatedChart

Description automatically generatedGraphical user interface, application

Description automatically generated The confusion matrixs above illustrates how ineffective the Adadelta optimiser was. In the case of VGG19, it was unable to categorise almost all of the other classifications. The 90% accuracy for happy and 30% accuracy for neutral accounts for the validation accuracy of approximately 52%. This mistake is present in all models but is not that pronounced in ResNet101V2.

Figure 57.ResNet101V2-Confusion matrix-Adadelta

Figure 58.InceptionV2-Confusion matrix-Adadelta

Figure 59.MobileNetV2-Clasification report-Adadelta

Figure 60.InceptionV3-Clasification report-Adadelta

Figure 61.ResNet101V2-Clasification report-Adadelta

Figure 62.VGG19-Clasification report-Adadelta

Figure 63.MobileNetV2-Confusion matrix-Adadelta

Figure 64.VGG19-Confusion matrix-Adadelta

According to the categorisation reports above, the accuracy and recall are much lower than those of the Adagrad. The VGG19 cannot have an F1-score greater than 0 for anger, disgust, fear, sadness, or surprise since their actual positive rate is 0. The other models outperformed the VGG19 in terms of generalisation, but they also had a recall of 0 for disgust. The other classes achieved poorer accuracy, recall, and F1-score enhancement than the Adagrad and Adam optimisers.

Adagrad may be influenced by the initial conditions and gradients of the parameters. When initial gradients are large, future learning rates are low. This may be addressed by increasing the global learning rate and tying the Adagrad method to the chosen learning rate. Additionally, because of the constant accumulation of squared gradients in the denominator, the learning rate decreases throughout training, eventually approaching 0 and ending completely. Zeiler (2017) invented the Adadelta method to circumvent hyper-parameter selection sensitivity and prevent learning rates from declining indefinitely. While this method may be superior in certain circumstances, it was unable to surpass Adagrad in this instance.

## 5.4. Fourth configuration models

Since Adam showed to be the most covetable choice, in this stage, it will be tuned by changing the values of the parameter called the learning rate. There were chose three learning rates, 0.5,0.01 and 0.001 values that were suggested by (Cs231n.github.io. ,2021), (Nielsen,2015)

Figure 65.Learning rate comparison for each model

Chart, line chart

Description automatically generatedChart, line chart

Description automatically generatedTable

Description automatically generatedApplication, table

Description automatically generated with medium confidence This experiment shows that the optimum learning rate is very architecture dependent. For instance, a score of 0.5 was not ideal for VGG19 InceptionV3, ResNet101V2, and MobileNetV2, which all scored 0.4741. Additionally, this value was acquired in the first period of training and remained constant throughout the programme. In comparison, the value 0.01 aided each model in developing its generalisation ability. Finally, because the 0.001 learning rate was the most effective in all cases, it will be utilised in future trials.

Figure 66.InceptionV3-learnng rate 0.5

Figure 67.MobileNetV2-learning rate 0.5

Figure 68.ResNet101V2-learning rate 0.5

Figure 69.VGG19-lerarnig rate 0.5

Calendar

Description automatically generatedCalendar

Description automatically generatedCalendar

Description automatically generatedCalendar

Description automatically generatedThe figures above illustrate the incapacity of the models to learn since the accuracy line for training a validation did not change from the first epoch. The cross-entropy function indicated a very large validation loss, which means that the models made many errors.

Figure 70.ResNet101V2-Confusion matrix-learning rate 0.5

Figure 71.InceptionV3-Confusion matrix-learning rate 0.5

Figure 72.MobileNetV2-Confusion matrix-learning rate 0.5

Figure 73.VGG19-Confusion matrix-learning rate 0.5

Table

Description automatically generatedTable

Description automatically generatedTable

Description automatically generatedTable

Description automatically generated According to the confusion matrix, all of the pictures were categorised as happy. This demonstrates that the 47% accuracy does not necessarily imply that the models improved throughout training. This rate of learning is incorrect for this kind of categorisation since no model is generalisable. The models obtained such high accuracy even though six out of seven classes have no accuracies because the happy class comprises 47% of all testing images.

Figure 74.InceptionV3-Clasification report-learning rate 0.5

Figure 75.MobileNetV2-Clasification report-learning rate 0.5

Figure 76.ResNet101V2-Clasification report-learning rate 0.5

Figure 77.VGG19-Clasification report-learning rate 0.5

Graphical user interface, application

Description automatically generatedGraphical user interface, application

Description automatically generated As with the confusion matrix, the only class with 1 speeking of recall is happy, where the recall value represents the number of images correctly categorised as happy and the precision value represents the percentage of images labelled as happy that really are.

Figure 78.ResNet101V2-learning rate 0.01

Figure 79.VGG19-learning rate 0.01

Graphical user interface, application, table

Description automatically generatedGraphical user interface, application

Description automatically generated

Figure 80.MobileNetV2-learning rate 0.01

Figure 81.InceptionV3-learning rate 0.01

Graphical user interface, application, table

Description automatically generatedGraphical user interface, table

Description automatically generatedGraphical user interface, chart

Description automatically generated The learning rate of 0.01 showed an improvement from the previous one. Even so, it is barely perceptible that the accuracy is continuously but very slowly improving. This is apparent in the slope of the accuracy diagram, which becomes higher at a minimal angle.

Figure 82.InceptionV3-Confusion matrix-learning rate 0.01

Figure 83.ResNet101V2-Confusion matrix-learning rate 0.01

Figure 84.VGG19-Confusion matrix-learning rate 0.01

Graphical user interface, application, table

Description automatically generatedHere the learning rate affected each model differently. The VGG19 struggles to identify other classes besides happy and neutral. It still has an accuracy of 18% for the surprise class, but it also classifies 18% of fear as surprise. The ResNet101V2 usually had the best result in each phase, but the overall accuracy is because of the happy and neutral classes. Still, it can match 16% of anger pictures, but it shows a new tendency to match the other classes like disgust, fear, sadness and surprise as anger in a proportion of 13 to 21%. The MobileNetV2 had the worst generalisation power since happy and neutral could not identify any of the other classes. The InceptionV3 follows the same pattern as the VGG19 but with higher accuracy for the sadness class.

Figure 85.MobileNetV2-Confusion matrix-0.01

Table

Description automatically generated Even if the VGG19, MobileNetV2, and InceptionV3 had similar validation accuracy, the classification report shows how different they act when classifying the pictures. The MobileNetV2 had 0% precision and recall on anger, happiness, fear, sadness, and surprise, while ResNet had 0% for disgust, fear, sadness, and surprise. Speaking of the generalisation capacity, VGG19 can recognise pictures belonging to more classes than any other model, even if it has a lower accuracy than the ResNet101V2..

## Table Description automatically generatedTable Description automatically generatedTable Description automatically generated5.5. Fifth configuration models

Figure 86.InceptionV3-Clasification report-learning rate 0.01

Figure 87.VGG19-Clasification report-learning rate 0.01

Figure 88.MobileNetV2-Clasification report-learning rate 0.01

Figure 89.ResNet101V2-Clasification report-learning rate 0.01

To further increase the models’ ability to generalise and their accuracy, new convolutional layers were added. Because all the convolutional layers are “frozen” in order to preserve the weights from ImageNet cannot be trained. After some experiments, two convolutional layers with a filter size of 3\*3 were added. A filter with a larger size may be too harsh and could make the neurons lose some information (Albawi, Mohammed, & Al-Zawi, 2017). The padding chosen was “same” to calculate and apply the padding necessary to the input picture in order for the output to have the same shape as the input. Moreover, a new technique called batch normalisation was applied. This decreased the number of epochs needed to train the network and applied some regularisation.

Figure 90.Training results for each model in the fifth stage

Graphical user interface, application

Description automatically generatedGraphical user interface, application, email

Description automatically generatedGraphical user interface, application

Description automatically generatedGraphical user interface, application

Description automatically generatedThe experiment of adding a new convolutional layer was a success. It slightly increased the accuracy of each model. The MobileNetV2 and ResNet101V2 benefitted the most by gaining 2% accuracy. Still, the ResNet101V2 produced the best results compared to all the other models.

Figure 91.InceptionV3-fifth stage

Figure 92.MobileNetV2-fifth stage

Figure 93.ResNet101V2-fifth stage

Figure 94.VGG19-fifth stage

The new layers added changed the learning path of each model. It can be seen that after adding the convolutional layers, the models started to overfit at some point. Also, the learning process becomes much more irregular because the validation accuracy tends to grow up to some point where it reaches a peak and then decreases. During training, there might be more spikes like that, showing the point where the model reached the maximum potential.

Table

Description automatically generatedTable

Description automatically generatedTable

Description automatically generatedTable

Description automatically generatedTable

Description automatically generated with medium confidenceTable

Description automatically generatedGraphical user interface, application

Description automatically generatedGraphical user interface, table

Description automatically generatedAt this stage, each of the models started to develop more highlighted diagonal. An error that each model tends to make is to classify the disgust class as happy or neutral. Therefore, it never achieves an accuracy of more than 6%. Furthermore, approximately 12 to 18% of pictures belonging to the fear class are classified. This behaviour can be seen in all architectures. The VGG19 and MobileNetV2 shear the problem of classifying the fear class as sadness in a proportion of 10 to 12%.

Figure 95.InceptionV3-Clsification report-fifth stage

Figure 96.MobileNetV2-Clasification report-fifth stage

Figure 97.ResNet101V2-Clasification report-fifth stage

Figure 98.VGG19-Clasification report-fifth stage

Figure 99.InceptionV3-Confusion matrix-fifth stage

Figure 100.MobileNetV2-Confusion matrix-fifth stage

Figure 101.ResNet101V2-Confusion matrix-fifth stage

Figure 102.VGG19-Confusion matrix-fifth stage

The classification report reveals that none of the models acquired an F1-score of 0% for the first time. In terms of precision, each of the models scored decent values of over 40%. VGG19 was the only one that had an accuracy under 40% for the sadness class. The recall values seem to be very close in all the models. VGG19 is still struggling with anger and fear showing a smaller recall. This means that the number of pictures belonging to those classes was correctly predicted in a tiny amount compared to the total number of pictures. Even if ResNet101V2 had a slightly smaller F1 score for disgust, it could achieve the best generalisation.

## 5.6. Sixth configuration models

From the previous experiments, the models started to overfit after the convolutional layers were added. To minimise the accuracy and loss difference between training and validation, two types of regularisations will be added: a dropout of 20% after the pooling layer and L2 regularization for each convolutional and fully connected layer. For now, the regularization rate for L2 used the default value of 0.001.

Figure 103.Training results for each model in the sixth stage

Graphical user interface, application

Description automatically generatedChart, line chart

Description automatically generatedIn the diagram, the performance of InceptionV3, ResNet101V2 and MobileNetV2 decreased after applying the regularization whit approximative 2%-3%. It seems that the VGG19 had to benefit because of this approach in terms of accuracy by achieving 4% in plus. The usage of the regularization along with the ModelCheckpoint helped to obtain the best variant with the smallest overfitting. This can be seen by the small difference between the training and validation values.

Figure 104.ResNet101V2-sixth stage

Figure 105.VGG19-sixth stage

ResNet101V2-sixth stage

VGG19-sixth stage

Chart, line chart

Description automatically generatedChart, line chart

Description automatically generated

Figure 106.InceptionV3-sixth stage

Figure 107.MobileNetV2-sixth stage

Graphical user interface, table

Description automatically generatedGraphical user interface, table

Description automatically generated with medium confidenceThe diagrams above show the benefits of using regularisation when the model is overfitting. ResNet101V2, MobileNetV2, and InceptionV3 had a better training process in the sixth stage than in the fifth stage. The orange and blue lines are united most of the time. The VGG19 showed signs of underfitting because the validation accuracy went across the training accuracy. This problem was solved in the next stage of development.

Figure 108.ResNetV2-Confusion matrix-sixth stage

Figure 109.VGG19-Confusion matrix-sixth stage

Table

Description automatically generatedGraphical user interface, table

Description automatically generated with medium confidence

Figure 110.IncptionV3-Confusion matrix-sixth stage

Figure 111.MobileNetV2-Confusion matrix-sixth stage

The regularisation helped with the overfitting problem but affected the method of generalising the other classes. The VGG19 increased the capacity to recognise the pictures belonging to the anger and happiness classes. Additionally, it distinguishes the happiness class from the others better than it did before. In ResNet101V2, the classes that had better accuracy in the previous stage, such as anger, fear, sadness, and surprise, were recognised as neutral. MobileNetV2 and InceptionV3 could not detect any significant difference between happy, neutral, and the rest of the classes.

Table

Description automatically generatedTable

Description automatically generatedTable

Description automatically generatedTable

Description automatically generated Regularisation significantly affected each model’s accuracy and recall, which took a value of 0%. Additionally, the F1 score decreased for all classes in all models. Thus, the present regularisation’s parameters impacted all classes somehow, but this may be rectified in the next step

Figure 112.Inceptionv3-Clasification report-sixth stage

Figure 113.MobileNetV2-Clasification report-sixth stage

Figure 114.ResNet101V2-Clasification report-sixth stage

Figure 115.VGG19-Clasification report-sixth stage

## 5.7. Seventh configuration models

The architecture of each model is different, so the setting of the parameters had to be chosen to match each model’s needs. In this stage, a wider hyper-tuning of the parameters using Keras Tuner occurred. The code developed was supposed to find the right number of convolutional layers, the fully connected layer, neurons, the dropout rate, and the regularisation rate. There was a total of 20 trials for each model where different values were used. The number of epochs set was 15, but early stopping was used to reduce the computational time by stopping the current training if the validation loss did not improve. The range of the values was carefully chosen to ensure that there was no way to exceed the resources required to compile the code. The code consisted of choosing between one and three convolutional layers. Each of these has a number of neurons between 32 and 512 with a default value of 128. For each convolutional layer, an L2 regularisation was added, which could take values between 0.01 and 0.0001. The dropout layer, which was the position after the global max pooling, had a dropout rate between 0 and 0.2. After the parameters were transformed into a 1d feature vector, the fully connected layers were added in a range of 2. These dense layers also had L2 regularisation that could take values between 0.01 and 0.0001. At the end, the final dense layer with the softmax activation was added. The optimisation approach was Adam with a learning rate of 0.001 since the experiments conducted previously showed that this is the best one. The best parameters found were then used to train the model.

The number of epochs for ResNet101V2 increased from 10 to 50, but no signs of improvement appeared after epoch 30. Therefore, MobileNetV2, InceptionV3, and VGG19 were trained on 30 epochs. Due to the high complexity of the VGG19, the compiling time for one epoch was over six hours, while for the other models it was approximately one hour.

Figure 116.Training results for each model in the seventh stage

Graphical user interface, application

Description automatically generatedGraphical user interface, chart, application, line chart

Description automatically generatedAfter hyper-tuning, each model increased its accuracy. The greatest difference can be seen in the VGG19, which improved by approximately 10% compared to the previous stage. Even so, the ResNet101V2 scored the highest in this stage as well. The diagram also shows that the saved models did not overfit since the training and validation values are so close.

Figure 117.ResNet101V2-seventh stage

Figure 118.VGG19-seventh stage

Graphical user interface, application, table

Description automatically generatedChart, line chart

Description automatically generatedTable

Description automatically generated with low confidenceTable

Description automatically generatedTable

Description automatically generated with low confidenceTable

Description automatically generated with medium confidence In this stage, the training process tended to be irregular. This is because of the spikes created by the validation accuracy. Fortunately, these spikes were not very pronounced because of the regularisation applied to minimise the overfitting.

Figure 119.InceptionV3-seventh stage

Figure 120.MobileNetV2-seventh stage

Figure 121.InceptionV3-Confusion matrix-seventh stage

Figure 122.MobileNetV2-Confusion matrix-seventh stage

Figure 123.ResNet101V2-Confusion matrix-seventh stage

Figure 124.VGG19-Confusion matrix-seventh stage

All the models were able to obtain a better generalisation for each class and develop the diagonal path in the matrix. Even so, besides the ResNet101V2, none of the models were able to classify the disgust class with over 10% accuracy. Additionally, the pictures were not distributed in such a great number as before in the happy and neutral classes. For example, over 20% of the fear class pictures were classified as sadness and surprise. Furthermore, the models show a pattern of classifying disgust as anger.

Table

Description automatically generatedTable

Description automatically generatedA picture containing text, indoor

Description automatically generatedA picture containing text, indoor

Description automatically generated According to the classification report, almost all classes in each model may achieve more than 50% precision. Fear and disgust were reduced to less than 40% in MobileNetV2 and InceptionV3, respectively. ResNet101V2 achieved a high recall for each class, resulting in an f1 score that outperformed all other models in every instance.

Figure 125.MobileNetvV2-Clasification report-seventh stage

Figure 126.InceptionV3-Clasification report-seventh stage

Figure 127.ResNet101V2-Clasification report-seventh stage

Figure 128.VGG19-Clasification report-seventh stage

## 5.8. Eighth configuration models

The ResNet101V2 demonstrated the best capacity for generalisation, which is why it was chosen for the next experiment. At this point, the architecture of the model was not further changed. Instead, the images were allocated differently. Until now, the training pictures were divided into 80% for training and 20% for validation. At this point, the whole dataset was used for training, and a new dataset was used for validation and testing. The validation dataset consists of 3,500 pictures equally divided among the classes

Figure 129.Training results for ResNet101V2 in the eighth stage

Chart, line chart

Description automatically generatedThe model had a very difficult time speaking of overfitting. Figure 129 represents the model’s performance at the best point. By using the new validation dataset as testing, the model reached a validation accuracy of 46.53%. However, even if the training accuracy went over 70%, the model was not able to generalise.

Figure 130.ResNet101V2-eighth stage

Graphical user interface

Description automatically generated with medium confidence Figure 130 shows that the model was overfitting from the beginning of the training. The training process was also irregular, and the best result was achieved in the most prominent spike. The regularisation applied also had an important role in preventing overfitting, but it seems that it was not enough. Therefore, if the regularisation had not been applied, the result could have been much worse.

Figure 131.ResNet101V2-Confusion matrix-eight stage

Table

Description automatically generatedFigure 131 illustrates how the equally-sized images for each class were distributed throughout the prediction process. The primary problem seems to be the model’s inability to correctly identify the disgust category. This is often confused with the anger class, which accounts for 38% of the population. The remaining 57% of images are divided into happy, neutral, and sad categories. Surprise is another category in which the inaccuracy is greater than the accuracy. While surprise is accurate to within a margin of error of 29%, surprise categorised as neutral has a margin of error of 33%. Additionally, a significant portion of the images from all other groups tend to be categorised as neutral.

Figure 132.ResNet101V2-Clasifiation report-eighth stage

Using a balanced testing set showed that although accuracy is often more than 50% for most classes, the recall has significant faults. For example, the model recognises a limited number of images from the surprise and fear categories and just a handful from the disgust category. This had a detrimental effect on the F1 score for those classes and the model’s overall performance..

# Evaluation and conclusion

# Chapter 6. Evaluation

This chapter is divided into two sections: 1) discussion of the product and 2) project evaluation. The first section contains a critical evaluation of the best models developed parallel with similar approaches. In the second section, the development of the entire project is discussed, including unforeseen problems that were solved. This section also highlights what objectives were achieved and what Constructing a machine learning model using deep CNNs was a difficult but fascinating adventure. Through this effort, many scholars such as Geoffrey Hinton, Alex Krizhevsky, Andrew Ng, Yoshua Bengio, and Yann LeCun contributed innovative insights that improved the system and simplified complex concepts. During the development process, planning, training, and testing, the DCNN proved challenging due to the abundance of literature and the process for constructing the best model not being as straightforward as just changing the input data with the new dataset utilised in this research. Indeed, using the same parameters for two distinct datasets creates inconsistent results.

During the design process, the literature offered a plethora of possible solutions. Academics such as Alex Krizhevsky, Yann LeCun, and Karen Simonyan have all contributed to these options. However, the primary constraint on adopting various types of architectures was training time, which grew as various variables such as the number of pictures or convolutional layers did. In some instances, these designs needed days or even weeks of training to reach acceptable performance levels. Indeed, there is a trade-off between network complexity and computation cost. The more layers, neurons, or filters a network has, the longer the processing time. As a result, it is critical to determine whether increasing the number of components is worth the additional processing time, but it is also necessary to remember that adding additional components does not ensure that the network’s performance will improve.

During the training step, hyper-parameters were selected by trial and error from various suggested values or techniques. However, since these suggestions were made for other types of datasets than the one utilised in this research, the experimental findings in some instances differed from what was published in the literature. To the extent feasible, all experiments were conducted by changing just one element at a time, notably hyper-parameters, input data, or architecture. Nevertheless, particular training processes required more than one change to reduce the computational time and make the algorithm more efficient.

The next step, testing, was conducted utilising 20% of the training set that was not utilised throughout the training process. Following this, a model was trained using the best hyper-parameters found throughout the experiments, and the model’s performance was evaluated using the test dataset. This whole procedure proved to be time-intensive since any error might influence subsequent trials. For instance, if the experiment determining the gradient descent optimisation method were conducted incorrectly, the experiment determining the model’s learning rate had to be redone.

After different stages of development for the best architecture, the best accuracy provided was 46.53% for seven emotions. The last stage of the development was conducted to compare the result achieved by the best model with the best models available, which also used the AffectNet dataset.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Model** | **Training data** | **Accuracy of 7 emotion** | **Paper** | **Year** |
| Emotion-GCN | Manually annotated AffectNet | 66.34 | [Exploiting Emotional Dependencies with Graph Convolutional Networks for Facial Expression Recognition](https://paperswithcode.com/paper/exploiting-emotional-dependencies-with-graph) | 2021 |
| Multi-task EfficientNet-B0 | AFEW+VGAF+AffectNet | 65.74 | [Facial expression and attributes recognition based on multi-task learning of lightweight neural networks](https://paperswithcode.com/paper/facial-expression-and-attributes-recognition) | 2021 |
| Distilled student | Google Facial Expression Comparison+AffectNet | 65.4 | [Leveraging Recent Advances in Deep Learning for Audio-Visual Emotion Recognition](https://paperswithcode.com/paper/leveraging-recent-advances-in-deep-learning) | 2021 |
| PAENet | FotW/Chalearn+VGGFace2+AffectNet | 65.29 | [Increasingly Packing Multiple Facial-Informatics Modules in A Unified Deep-Learning Model via Lifelong Learning](https://paperswithcode.com/paper/increasingly-packing-multiple-facial) | 2019 |
| ARM | RAF-DB+ SFEW+ AffectNet | 65.2 | [Learning to Amend Facial Expression Representation via De-albino and Affinity](https://paperswithcode.com/paper/learning-to-amend-facial-expression) | 2021 |
| DACL | AffectNet | 65.2 | [Facial Expression Recognition in the Wild via Deep Attentive Center Loss](https://paperswithcode.com/paper/facial-expression-recognition-in-the-wild-via) | 2021 |
| CPG | VGGFace2+AffectNet | 63.57 | [Compacting, Picking and Growing for Unforgetting Continual Learning](https://paperswithcode.com/paper/compacting-picking-and-growing-for) | 2019 |
| CAKE | AffectNet | 61.7 | [CAKE: Compact and Accurate K-dimensional representation of Emotion](https://paperswithcode.com/paper/cake-compact-and-accurate-k-dimensional) | 2018 |
| Facial Motion Prior Network | AffectNet | 61.52 | [Facial Motion Prior Networks for Facial Expression Recognition](https://paperswithcode.com/paper/facial-motion-prior-networks-for-facial) | 2019 |
| CNNs and BOVW+local SVM | AffectNet | 59.58 | [Local Learning with Deep and Handcrafted Features for Facial Expression Recognition](https://paperswithcode.com/paper/local-learning-with-deep-and-handcrafted) | 2019 |
| **ResNet101V2 fine-tuned** | **AffectNet** | **46.53** | **Machine Learning-based Automated Facial Emotion Recognition** | **2021** |

Table 7.Models comparison

Table 7 shows that none of the developers have managed to produce excellent results. This is because of the AffecteNet dataset, which is a very challenging database containing pictures of people of different ethnicities and ages (Antoniadis, Filntisis, & Maragos, 2021). Furthermore, according to the database’s creators, it is “heavily unbalanced” (Mollahosseini et al., 2019). In this scenario, classifiers often create a biased learning model with worse prediction accuracy for minority classes than for majority classes. This is because most commonly used classifier learning algorithms such as decision trees, backpropagation neural networks, and SVMs are designed to assume that the class distribution is relatively balanced and the cost of misclassification is equal (Zheng & Jin, 2020). This explains the low accuracy of the other classes since the happy and neutral classes constitute approximately 70% of the whole dataset. The main challenge of the algorithm was to create a very robust architecture capable of learning and distinguishing the classes even if they had a small number of pictures. However, given the existing resources and the limited approaches available to satisfy the resource requirements, the model reached a reasonable accuracy.

All of the models mentioned in the table used transfer learning and image augmentation to achieve a reasonable generalisation rate. The difference was that some of the architectures were developed using more complex and deeper neural networks like DenseNet and EficientNet (Antoniadis et al., 2021; Savchenko, 2021). Other academics like Shi et al. (2021), Farzaneh and Qi (2021), Hung et al. (2019), and Kervadec et al. (2018) chose a simpler architecture like ResNet18 or ResNet50, which were trained from scratch on a dataset for face recognition like VGGFace2 or MS-CELEB-1M which have over 3 million pictures. The weights from ImageNet were then added to the models that were already trained on the aforementioned dataset. Moreover, Savchenko (2021), Schoneveld et al. (2021), Hung et al. (2019), and Shi et al.(2021) tried to solve the problem of the unbalanced dataset and added more pictures to the classes that were outnumbered. This was done by combining other available datasets, which are mentioned in the table 7. Given the conditions that were available to create the fine-tuned ResNet101V2, the experiments were a success. There is still enough room for improvement, which can be achieved by adopting more of the approaches described above.

## 6.2. Project evaluation

Facial emotion recognition is a difficult job that requires an extensive understanding of psychology and artificial intelligence. Once the project’s concept was selected, research into those areas began immediately. Apart from the fundamentals of emotion recognition, machine learning, and deep learning, it was necessary to master the Python programming language. A strategy was established at the project’s inception to ensure that sufficient time was allotted for studying, coding, and writing the dissertation. The institution supplied instructional materials, online courses, a supervisor, and research materials. Regrettably, some roadblocks prolonged the procedure. The processing time was more significant than anticipated, and the resources used by the entity did not allow for the simultaneous execution of numerous activities required by other modules. As a result, the procedure had to be suspended.

The initial plan developed appropriately worked until the moment the code needed to be halted. This happened when another project needed to be finished and submitted. This delayed the whole process by up to two months. The only change that could have been made would have been to borrow a laptop from the university powerful enough to run the other projects on it while the code for the dissertation was running on the personal laptop.

The project’s initial goals were effectively achieved. First, the researcher read all the necessary material to grasp the basics of machine learning and deep learning. This was also feasible as a result of the university’s many online courses and modules. The methods needed to enhance the models’ performance were found by reviewing many papers on machine learning, deep learning, and facial expression detection. Additionally, these resources offered advice for determining and comprehending which software, tools, and libraries should be used to build the algorithm. Pre-trained models were carefully selected based on publications and complexity to fine-tune them without crashing the machine. The end product’s generalisation rate was compared to some of the top models that utilise the same dataset.

The primary disadvantage encountered was the number of resources required. Throughout the development phases, the most frequently encountered issue was computer crashing. This was due to the RAM being utilised at 100%, which caused the machine to shut down. Unfortunately, this option was also available for the GPU’s V-RAM memory. Therefore, it was necessary to determine the optimal settings that matched the GPU and RAM requirements to perform the tests. Another unexpected issue was the CPU temperature, which sometimes exceeded 100 degrees, causing the machine to shut down as a safety precaution. This occurred because of the lengthy period required to train the models, which in some instances exceeded two weeks.

Google Colab Pro was particularly challenging to handle due to the disconnections that occurred every hour and a half while the web page was idle. Additionally, it prohibited notebooks from operating for more than 12 hours continuously and in certain instances even less depending on the amount of use. In order to use these tools, some calculation had to be made to ensure that the experiment would not exceed the total amount of time allowed and that the code was checked every hour to ensure that it had not been disconnected.

Some ethical, legal, and social measures were considered when developing the code. If the code were to be implemented in a surveillance application, it is critical to avoid discrimination based on skin colour and ethnicity (Rhue, 2021). After testing multiple facial emotion recognition algorithms, the programs assigned more negative emotions like anger to African people. This can be avoided by using a diverse dataset with different ages and ethnicities. This is also applicable for persons with medical conditions and physical impairments, which can cause a wide range of misclassification (Horvath & Vemou, 2021).

Furthermore, using facial emotion recognition remotely does not allow people to provide consent regarding how their data can be used. For example, by analysing facial expressions, computers may identify indications of alexithymia, a condition in which an individual is unable to comprehend their own emotions or lacks the vocabulary to explain them. This discovery may have implications for severe mental and neurological diseases such as psychosis. This kind of information can be made public and can affect the image of public figures like politicians and celebrities (Horvath & Vemou, 2021). Facial emotion recognition can also be used in applications to manipulate vulnerable people at that point and force them to commit actions that they would not normally engage in, like buying goods they do not need (Horvath & Vemou, 2021).

# Chapter 7. Conclusion and recommendation

This dissertation was intended to develop an automated system for facial emotion recognition via the use of machine learning and deep learning methods. This was accomplished by training and testing several deep convolutional network designs and fine-tuning the hyper-parameters to arrive at the optimal mix. The dissertation concentrated on seven critical phases in creating an automatic FER programme: data collection, pre-processing, system design, development, training, testing, and assessment. The most critical section of the dissertation was the background knowledge that influenced the project’s conception. Additionally, it included key concepts from other academics who specialise in machine learning and deep learning during training and testing.

The experimental part was conducted using the AffectNet dataset that had been manually annotated. Four distinct deep learning neural networks were suggested for training and testing. The code was developed in eight phases, each showing critical procedures for increasing the accuracy of the models and preventing them from overfitting or underfitting. It has been shown that layers with more complete connections and fewer parameters may achieve greater accuracy than layers with just one complicated layer. However, caution should be used with the settings since an incorrect combination of neurons may decrease accuracy. Selecting the appropriate optimiser also results in an improvement in performance. Here, the tests have shown that Adam outperformed the other candidates. The learning rate of 0.001 was shown to be the most appropriate for optimisers since the other evaluated rates decreased the model’s generalisability. Adding convolutional layers improved the accuracy further but resulted in overfitting the models. The issue was solved using two regularisation techniques: dropout and L2 regularisation. The most critical step of model creation was fine-tuning the parameters since a successful model can only be accomplished by finding the optimal balance of all the parameters. Among all the models, ResNet101V2 demonstrated the greatest generality, which is why it was trained in particular environments and assessed using comparable methods. This dissertation describes and records the project’s deliverables, which include the research approach, system implementation, experimental results, and their interpretation, all of which led to a deeper knowledge of DCNN.

There is still considerable potential for further development. One possibility is to evaluate other available architectures since some may perform better in particular situations or to build a new one from the start of this job. If sufficient resources are available, models should be trained for facial recognition using large datasets such as VGGFace2 or MS-CELEB-1M before being transferred to utilise the acquired knowledge. Additionally, these weights may be merged with those acquired by training the models on the ImageNet dataset. The dataset used to create facial emotion recognition should be balanced in terms of the number of images in each class, including a range of races and ages. Other methods must then be tested, such as evaluating additional gradient descent optimisers, learning rates for individual optimisers, combinations of regularisations, and other important parameters that require rigorous hyper-tuning.

# References

1. Chibelushi, C. and Bourel, F. ,2016. Facial Expression Recognition: A Brief Tutorial Overview.
2. Prasad, D., 2012. Survey of The Problem of Object Detection In Real Images. [online] Researchgate. Available at: <https://www.researchgate.net/publication/235216716\_Survey\_of\_The\_Problem\_of\_Object\_Detection\_In\_Real\_Images> [Accessed 6 June 2021].
3. Y. Lv, Z. Feng and C. Xu, 2014 "Facial expression recognition via deep learning," Smart Computing (SMARTCOMP), International Conference on, Hong Kong, 2014, pp. 303-308. doi: 10.1109/SMARTCOMP.2014.7043872
4. Liu, P., 2014. Facial Expression Recognition via a Boosted Deep Belief Network. 2014 IEEE Conference on Computer Vision and Pattern Recognition,.
5. Fasel, B. and Luettin, J., 2003. Automatic facial expression analysis: a survey. Pattern Recognition, 36(1), pp.259-275.
6. Ufldl.stanford.edu. ,2021. Unsupervised Feature Learning and Deep Learning Tutorial. [online] Available at: <http://ufldl.stanford.edu/tutorial/supervised/ConvolutionalNeuralNetwork/> [Accessed 15 Apr. 2021].
7. Cs231n.github.io. (2021). CS231n Convolutional Neural Networks for Visual Recognition. [online] Available at: http://cs231n.github.io/convolutional-networks/ [Accessed 15 Apr. 2021].
8. Bishop, C. ,2006. Pattern recognition and machine learning. New York: Springer, pp.227 - 249 and 256 - 272.
9. Murphy, K. ,2012. Machine learning. Cambridge, Mass.: MIT Press, pp. 563 - 579.
10. MacKay, D. ,2003. Information theory, inference, and learning algorithms. Cambridge, UK: Cambridge University Press, pp. 467 - 492.
11. Poczos, B. and Singh, A. ,2016. Introduction to Machine Learning. Deep Learning. CMU.
12. Goodfellow, I., Warde-Farley, D., Mirza, M., Courville, A. and Bengio, Y. ,2013. Maxout Networks. JMLR WCP 28, pp.1319-1327. arXiv:1302.4389 [stat.ML]
13. B. Liu, W. Shen, P. Li and X. Zhu, 2019 "Accelerate Mini-batch Machine Learning Training With Dynamic Batch Size Fitting," International Joint Conference on Neural Networks (IJCNN), 2019, pp. 1-8, doi: 10.1109/IJCNN.2019.8851944.
14. Burlutskiy, N., Petridis, M., Fish, A., Chernov, A. and Ali, N., 2016. An Investigation on Online Versus Batch Learning in Predicting User Behaviour. Research and Development in Intelligent Systems XXXIII, pp.135-149.
15. Ruder, S. ,2016. An overview of gradient descent optimisation algorithms. ArXiv, abs/1609.04747.
16. Zeiler, M. ,2017. Adadelta: An adaptive learning rate method. Google Inc., USA New York University, USA. arXiv:1212.5701 [cs.LG]
17. Hinton, G. ,2016. Lecture 6e. rmsprop: Divide the gradient by a running average of its recent magnitude. [online] Available at: <https://www.cs.toronto.edu/~hinton/coursera/lecture6/lec6.pdf>
18. Keras, 2021. Keras documentation: Optimizers. [online] Keras.io. Available at: <https://keras.io/api/optimizers/> [Accessed 31 July 2021].
19. Kingma, D. and Ba, J. ,2015. Adam: A method for Stochastic Optimisation. arXiv:1412.6980 [cs.LG]
20. Rodrigues, C. ,2015. Exploring the transfer learning aspect of deep neural networks in facial information processing. University of Manchester., p.23.
21. Hastie, T., Tibshirani, R. and Friedman, J. (2001). The elements of statistical learning. New York: Springer, pp.219-228.
22. Le Cun, Y., Denker, J. and Solla, S. (1990). Optimal Brain Damage. AT&T Bell Laboratories. ISBN:1-55860-100-7
23. Prechelt, L., 2012. Early Stopping — But When?. Lecture Notes in Computer Science, pp.53-67.
24. Wang, K., Guo, P., Luo, A., Xin, X. and Duan, F., 2016. Deep neural networks with local connectivity and its application to astronomical spectral data. 2016 IEEE International Conference on Systems, Man, and Cybernetics (SMC),.
25. Galeshchuk, S. and Mukherjee, S., 2017. Deep networks for predicting direction of change in foreign exchange rates. Intelligent Systems in Accounting, Finance and Management, 24(4), pp.100-110. Galeshchuk, S. and Mukherjee, S., 2017. Deep networks for predicting direction of change in foreign exchange rates. Intelligent Systems in Accounting, Finance and Management, 24(4), pp.100-110.
26. Goodfellow, I., Bengio, Y. and Courville, A., 2016. Deep learning. Cambridge (EE. UU.): MIT Press, pp.164-366.
27. Shocher, A., Feinstein, B., Haim, N. and Irani, M., 2021. From Discrete to Continuous Convolution Layers. [online] Arxiv.org. Available at: <https://arxiv.org/pdf/2006.11120.pdf> [Accessed 19 July 2021].
28. Ying, X., 2019. An Overview of Overfitting and its Solutions. Journal of Physics: Conference Series, 1168, p.022022.
29. Rao, N., Prasad, V., Teja, S., Zindavali and Reddy, P., 2018. A Survey on Prevention of Overfitting in Convolution Neural Networks Using Machine Learning Techniques. International Journal of Engineering & Technology, 7(2.32), p.177
30. Shorten, C. and Khoshgoftaar, T., 2019. A survey on Image Data Augmentation for Deep Learning. Journal of Big Data, 6(1)..
31. Mollahosseini, A., Chan, D. and Mahoor, M., 2016. Going deeper in facial expression recognition using deep neural networks. 2016 IEEE Winter Conference on Applications of Computer Vision (WACV),.
32. Lopes, A., de Aguiar, E., De Souza, A. and Oliveira-Santos, T., 2017. Facial expression recognition with Convolutional Neural Networks: Coping with few data and the training sample order. Pattern Recognition, 61, pp.610-628.
33. Mohammadpour, M., Khaliliardali, H., Hashemi, S. and AlyanNezhadi, M., 2017. Facial emotion recognition using deep convolutional networks. 2017 IEEE 4th International Conference on Knowledge-Based Engineering and Innovation (KBEI),.
34. Cai, J., Chang, O., Tang, X., Xue, C. and Wei, C., 2018. Facial Expression Recognition Method Based on Sparse Batch Normalization CNN. 2018 37th Chinese Control Conference (CCC),.
35. Li, Y., Zeng, J., Shan, S. and Chen, X., 2019. Occlusion Aware Facial Expression Recognition Using CNN With Attention Mechanism. IEEE Transactions on Image Processing, 28(5), pp.2439-2450.
36. Yolcu, G., Oztel, I., Kazan, S., Oz, C., Palaniappan, K., Lever, T. and Bunyak, F., 2019. Facial expression recognition for monitoring neurological disorders based on convolutional neural network. Multimedia Tools and Applications, 78(22), pp.31581-31603.
37. Agrawal, A. and Mittal, N., 2019. Using CNN for facial expression recognition: a study of the effects of kernel size and number of filters on accuracy. The Visual Computer, 36(2), pp.405-412.
38. Jain, D., Shamsolmoali, P. and Sehdev, P., 2019. Extended deep neural network for facial emotion recognition. Pattern Recognition Letters, 120, pp.69-74.
39. Kim, D., Baddar, W., Jang, J. and Ro, Y., 2019. Multi-Objective Based Spatio-Temporal Feature Representation Learning Robust to Expression Intensity Variations for Facial Expression Recognition. IEEE Transactions on Affective Computing, 10(2), pp.223-236.
40. Yu, Z., Liu, G., Liu, Q. and Deng, J., 2018. Spatio-temporal convolutional features with nested LSTM for facial expression recognition. Neurocomputing, 317, pp.50-57.
41. Liang, D., Liang, H., Yu, Z. and Zhang, Y., 2019. Deep convolutional BiLSTM fusion network for facial expression recognition. The Visual Computer, 36(3), pp.499-508.
42. Schoneveld, L., Othmanib, A. and Abdelkawya, H., 2021. Leveraging Recent Advances in Deep Learning for Audio-Visual Emotion Recognition. [online] Arxiv.org. Available at: <https://arxiv.org/pdf/2103.09154v1.pdf> [Accessed 2 August 2021].
43. Shi, J., Zhu, S. and Liang, Z., 2021. Learning to Amend Facial Expression Representation via De-albino and Affinity. [online] Arxiv.org. Available at: <https://arxiv.org/pdf/2103.10189v2.pdf> [Accessed 1 August 2021].
44. Savchenko, A., 2021. Facial expression and attributes recognition based on multi-task learning of lightweight neural networks. [online] Arxiv.org. Available at: <https://arxiv.org/pdf/2103.17107.pdf> [Accessed 2 August 2021].
45. Pourmirzaei, M., Esmaili, F. and Montazer, G., 2021. Using Self-Supervised Co-Training to Improve Facial Representation. [online] Arxiv.org. Available at: <https://arxiv.org/ftp/arxiv/papers/2105/2105.06421.pdf> [Accessed 2 August 2021].
46. Vo, T., Lee, G., Yang, H. and Kim, S., 2020. Pyramid With Super Resolution for In-the-Wild Facial Expression Recognition. IEEE Access, 8, pp.131988-132001.
47. Mollahosseini, A., Hasani, B. and Mahoor, M., 2019. AffectNet: A Database for Facial Expression, Valence, and Arousal Computing in the Wild. IEEE Transactions on Affective Computing, 10(1), pp.18-31.
48. Torrey, L. and Shavlik, J., 2010, n.d. Transfer Learning. Handbook of Research on Machine Learning Applications and Trends, pp.242-264.
49. Weiss, K., Khoshgoftaar, T. and Wang, D., 2016. A survey of transfer learning. Journal of Big Data, 3(1).
50. Lu, J., Behbood, V., Hao, P., Zuo, H., Xue, S. and Zhang, G., 2015. Transfer learning using computational intelligence: A survey. Knowledge-Based Systems, 80, pp.14-
51. He, K., Zhang, X., Ren, S. and Sun, J., 2015. Deep Residual Learning for Image Recognition. [online] Arxiv.org. Available at: <https://arxiv.org/pdf/1512.03385.pdf> [Accessed 4 August 2021].
52. ImageNet. 2021. Download ImageNet Data. [online] Available at: <https://www.image-net.org/download.php> [Accessed 2 August 2021].
53. Simonyan, K. and Zisserman, A., 2015. VERY DEEP CONVOLUTIONAL NETWORKS FOR LARGE-SCALE IMAGE RECOGNITION. [online] Arxiv.org. Available at: <https://arxiv.org/pdf/1409.1556.pdf> [Accessed 5 August 2021].
54. Szegedy, C., Vanhoucke, V., Ioffe, S. and Shlens, J., 2015. Rethinking the Inception Architecture for Computer Vision. [online] Arxiv.org. Available at: <https://arxiv.org/pdf/1512.00567.pdf> [Accessed 7 August 2021].
55. Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V. and Rabinovich, A., 2015. Going deeper with convolutions. 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR),.
56. Howard, A., Zhu, M., Chen, B., Kalenichenko, D., Wang, W., Weyand, T., Andreetto, M. and Adam, H., 2017. MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications. [online] Arxiv.org. Available at: <https://arxiv.org/pdf/1704.04861.pdf> [Accessed 6 August 2021].
57. NVIDIA Developer. 2021. Convolutional Neural Network (CNN). [online] Available at: <https://developer.nvidia.com/discover/convolutional-neural-network> [Accessed 8 August 2021].
58. Research.google.com. 2021. Colaboratory – Google. [online] Available at: <https://research.google.com/colaboratory/faq.html> [Accessed 8 August 2021].
59. Brownlee, J. (2014). Best Programming Language for Machine Learning - Machine Learning Mastery. [online] Machine Learning Mastery. Available at: http:// machinelearningmastery.com/best-programming-language-for-machine-learning/ [Accessed 8 August. 2021].
60. Aruoba, S. and Fernández-Villaverde, J., 2015. A comparison of programming languages in macroeconomics. Journal of Economic Dynamics and Control, 58, pp.265-273.
61. Anaconda. 2021. Anaconda | The World's Most Popular Data Science Platform. [online] Available at: <https://www.anaconda.com/> [Accessed 5 August 2021].
62. NVIDIA Developer. 2021. NVIDIA cuDNN. [online] Available at: <https://developer.nvidia.com/cudnn> [Accessed 9 August 2021].
63. The Official NVIDIA Blog. 2021. What Is CUDA | NVIDIA Official Blog. [online] Available at: <https://blogs.nvidia.com/blog/2012/09/10/what-is-cuda-2/> [Accessed 9 August 2021].
64. TensorFlow. 2021. TensorFlow. [online] Available at: <https://www.tensorflow.org/> [Accessed 9 August 2021].
65. Matplotlib.org. 2021. Pyplot tutorial — Matplotlib 3.4.2 documentation. [online] Available at: <https://matplotlib.org/stable/tutorials/introductory/pyplot.html> [Accessed 9 August 2021].
66. Numpy.org. 2021. NumPy. [online] Available at: <https://numpy.org/> [Accessed 9 August 2021].
67. Kim, H. and Jeong, Y., 2019. Sentiment Classification Using Convolutional Neural Networks. Applied Sciences, 9(11), p.2347.
68. Jeong, J., 2019. The Most Intuitive and Easiest Guide for CNN. [online] Medium. Available at: <https://towardsdatascience.com/the-most-intuitive-and-easiest-guide-for-convolutional-neural-network-3607be47480> [Accessed 9 August 2021].
69. Docs.python.org. 2021. os — Miscellaneous operating system interfaces — Python 3.9.6 documentation. [online] Available at: <https://docs.python.org/3/library/os.html> [Accessed 10 August 2021].
70. Scikit-learn.org. 2021. scikit-learn Tutorials — scikit-learn 0.24.2 documentation. [online] Available at: <https://scikit-learn.org/stable/tutorial/index.html> [Accessed 10 August 2021].
71. Nielsen, M. ,2015. Neural Networks and Deep Learning. Determination Press. [online] Available at: http://neuralnetworksanddeeplearning.com/chap3.html [Accessed 9 August 2021]
72. Ioffe, S. and Szegedy, C., 2015. Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. [online] arXiv.org. Available at: <https://arxiv.org/abs/1502.03167?source=post\_page-----901fd6e5393e----------------------> [Accessed 10 August 2021].
73. Happy, S. and Routray, A., 2015. Automatic Facial Expression Recognition Using Features of Salient Facial Patches. [online] Ieeexplore.ieee.org. Available at: <https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=6998925> [Accessed 14 August 2021].
74. Majumder, A., Behera, L. and Subramanian, V., 2018. Automatic Facial Expression Recognition System Using Deep Network-Based Data Fusion - IEEE Journals & Magazine. [online] Ieeexplore.ieee.org. Available at: <https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=7747479> [Accessed 14 August 2021].
75. Matsumoto, D., Hwang, H.S., 2011, Reading facial expressions of emotion, Psychological Science Agenda
76. Matsumoto Burkert, P., Trier, F., Afzal, M., Dengel, A. and Liwick, M., 2016. Depression: Deep Convolutional Neural Network For Expression Recognition. [online] Arxiv.org. Available at: <https://arxiv.org/pdf/1509.05371.pdf> [Accessed 14 August 2021].
77. Fasel, B. and Luettin, J., 2019. Automatic Facial Expression Analysis: A Survey. [online] Available at: <https://doi.org/10.1016/S0031-3203(02)00052-3> [Accessed 14 August 2021].
78. Zhang, L., Jiang, M., Farid, D. and Hossain, M., 2013. Intelligent facial emotion recognition and semantic-based topic detection for a humanoid robot. Expert Systems with Applications, 40(13), pp.5160-5168.
79. Lansley, J., 2020. Facial Expressions · EIA. [online] The Emotional Intelligence Academy. Available at: <https://www.eiagroup.com/knowledge/facial-expressions/> [Accessed 14 August 2020].
80. Wang, H. and Gu, J., 2018. The Applications of Facial Expression Recognition in Human-computer Interaction. 2018 IEEE International Conference on Advanced Manufacturing (ICAM).
81. Samadiani, N., Huang, G., Cai, B., Luo, W., Chi, C., Xiang, Y. and He, J., 2019. A Review on Automatic Facial Expression Recognition Systems Assisted by Multimodal Sensor Data. Sensors, 19(8), p.1863.
82. Mode Resources. 2021. Pandas | Python Library - Mode. [online] Available at: <https://mode.com/python-tutorial/libraries/pandas/> [Accessed 9 August 2021].
83. Albawi, S., Mohammed, T. and Al-Zawi, S., 2017. Understanding of a convolutional neural network. 2017 International Conference on Engineering and Technology (ICET),.
84. Antoniadis, P., Filntisis, P. and Maragos, P., 2021. Exploiting Emotional Dependencies with Graph Convolutional Networks for Facial Expression Recognition. [online] Arxiv.org. Available at: <https://arxiv.org/pdf/2106.03487v1.pdf> [Accessed 1 August 2021].
85. Farzaneh, A. and Qi, X., 2021. Facial Expression Recognition in the Wild via Deep Attentive Center Loss. 2021 IEEE Winter Conference on Applications of Computer Vision (WACV),.
86. Hung, S., Tu, C., Wu, C., Chen, C., Chan, Y. and Chen, C., 2019. Compacting, Picking and Growing for Unforgetting Continual Learning. [online] Arxiv.org. Available at: <https://arxiv.org/pdf/1910.06562v3.pdf> [Accessed 3 August 2021].
87. Kervadec, C., Vielzeuf, V., Pateux, S., Lechervy, A. and Jurie, F., 2018. CAKE: Compact and Accurate K-dimensional representation of Emotion. [online] arXiv.org. Available at: <https://arxiv.org/abs/1807.11215> [Accessed 4 August 2021].
88. Chen, Y., Wang, J., Chen, S., Shi, Z. and Cai, J., 2019. Facial Motion Prior Networks for Facial Expression Recognition. [online] Ieeexplore.ieee.org. Available at: <https://ieeexplore.ieee.org/document/8965826> [Accessed 4 August 2021].
89. Georgescu, M., Ionescu, R. and Popescu, M., 2019. Local Learning With Deep and Handcrafted Features for Facial Expression Recognition. IEEE Access, 7, pp.64827-64836.
90. Zheng, W. and Jin, M., 2020. The Effects of Class Imbalance and Training Data Size on Classifier Learning: An Empirical Study. SN Computer Science, 1(2).
91. Hung, S., Lee, J., Wan, T., Chen, C., Chan, Y. and Chen, C., 2019. Increasingly Packing Multiple Facial-Informatics Modules in A Unified Deep-Learning Model via Lifelong Learning. Proceedings of the 2019 on International Conference on Multimedia Retrieval,.
92. Rhue, L., 2021. Racial Influence on Automated Perceptions of Emotions. [online] Available at: <[http://dx.doi.org/10.2139/ssrn.3281765](https://dx.doi.org/10.2139/ssrn.3281765)> [Accessed 10 August 2021].
93. HORVATH, A. and VEMOU, K., 2021. Facial Emotion Recognition. [online] Op.europa.eu. Available at: <https://op.europa.eu/en/publication-detail/-/publication/709bd47b-be94-11eb-a925-01aa75ed71a1> [Accessed 15 August 2021].
94. Pourmirzaei, M., Esmaili, F. and Montazer, G., 2021. Using Self-Supervised Co-Training to Improve Facial Representation. [online] Arxiv.org. Available at: <https://arxiv.org/ftp/arxiv/papers/2105/2105.06421.pdf> [Accessed 19 July 2021].
95. Siqueira, H., Magg, S. and Wermter, S., 2020. Efficient Facial Feature Learning with Wide Ensemble-based Convolutional Neural Networks. [online] Arxiv.org. Available at: <https://arxiv.org/pdf/2001.06338v1.pdf> [Accessed 21 July 2021].
96. Ko, B., 2018. A Brief Review of Facial Emotion Recognition Based on Visual Information. Sensors, 18(2), p.401.

# Appendix A: Terms of reference (TOR)

Alexandru-Adonis Neagu

18017575

Computer Science

Supervisor: Dr Ossama Alshabrawy

Second marker: Dr Paul Vickers

**Background to Project**

Every human being has felt that in one way or another, they express these emotions as a way of communication. People can show our feelings in different ways as body language, voice tonality and facial expression. ( Happy, S. and Routray, A., 2015), (Majumder, A., Behera, L. and Subramanian, V., 2018).

           Recognising facial expression has been a subject that arouses the curiosity of many researchers and philosophers. In 1872 Charles Darwin was the first one who suggested that there are six universal emotions that humans can read and understand all over the world. (Matsumoto, D., Hwang, H.S., 2011) This suggestion has been rejected by many researchers such as Paul Ekman, Wallace Friesen and Phoebe Ellsworth (Matsumoto, D., Hwang, H.S., 2011). However, later Silvan Tomkins recruited Paul Ekman and Carroll Izard to continue Charles Darwin hypothesis, and they concluded that regardless of location and culture people can express and understand the following emotions: surprise, fear, disgust, anger, happiness and sadness. (Happy, S. and Routray, A., 2015), (Fasel, B. and Luettin, J., 2019)

           Ekman found that by reading facial emotion, we would be able to figure out patterns that can help us predict behaviours or thoughts. (Ekman, P. and Friesen, 1984) Nowadays, the advancing of technology opens new pathways in how we can use facial emotion recognition, and it has many applications in artificial intelligence domain.

           In the field of robotics, facial emotions play a massive role in human-computer interaction. In order to develop a robot that can simulate human behaviour, it has to learn by itself how to associate the inner feeling and the facial movements. ( Zhang, L. et al., 2013) A great example would be Sophia, a very advance robot that can learn and display facial emotions using artificial intelligence. ( https://www.hansonrobotics.com/sophia/)

           Teaching has changed lately, and e-learning is the most common way to teach due to the pandemic. The technology of facial emotion recognition can assist teachers by measuring the learning levels of the students, but it can also predict satisfaction and understanding when they participate in a lecture session. (Lansley, J., 2020), (Wang, H. and Gu, J., 2018)

           In the medical field, facial emotion recognition has begun to be used by doctors in the treatment of mental and psychiatric diseases. ( Samadiani, N. et al., 2019) The system could use a mix of facial emotion recognition, body recognition and voice tone recognition to expose subtitle information that will help doctors to make judgements on the emotional and mental state of people with autism, depression, schizophrenia and so on. (Lansley, J., 2020), (Wang, H. and Gu, J., 2018)

           Large companies are taking advantage of facial emotion recognition to increase their selling. They are collecting the facial expression of people when revelling a new product such as a phone or a car. Using this method, they can obtain a hones reaction when presenting the design and specification and predict whether the product will be profitable or not. Also, they are looking forward to developing a new way of recruiting trustworthy employees. Such a process implies a recorded questionnaire taken by the potential employees, which is then analysed using face emotion detection. (Wang, H. and Gu, J., 2018)

           Airports, military, security firms are also showing interest in researching the use of cameras and software to pair everything that is known about facial emotion recognition and deception in order detect any suspicious movements that can indicate drug traffic, terrorist attacks or any other potential crimes that can occur in a large and busy space. (Lansley, J., 2020),( Wang, H. and Gu, J., 2018)

           One of the reasons why I chose this subject is the growing interest that the companies are showing in this technology. I find this subject attractive because I find fascinating the idea that we can identify and predict the behaviour of human being. Moreover, I want to bring a small contribution to all the efforts made by other researchers and push myself to learn new stuff and develop my skills as a computer scientist.

**Data**

Automated affective computing within the wild setting is a challenging issue in computer vision. Existing annotated databases of facial expressions within the wild are little and mostly cover discrete emotions. There are exceptionally constrained commented on facial databases for personal computing within the continuous dimensional show. For this project will be used data from the AffectNet database. Since the database consists only in facial images, there will not be any other kind of data that will need to be collected. This database is by far the largest database of facial expressions, valence and in arousal in the wild enabling research in automated facial expression recognition in two different emotion models. It was created by collecting and annotating facial images. The word Affect is a mental term used to portray the outward articulation of feeling and emotions.

The AffectNet database contains about 1M facial pictures gathered from the internet by questioning three significant web search engines utilising 1250 emotions related watchwords in six different languages. There are approximative 550000 images that were automatically annotated using ResNext Neural Network. These samples were obtained by using the other samples as a training set. The other 420000 samples are manually annotated for the presence of seven discrete facial expressions and the intensity of valence and arousal. The database provides eleven annotated emotions: Neutral, Happiness, Sadness, Surprise, Fear, Disgust, Anger, Contempt, None, Uncertain and No-Face.

|  |  |
| --- | --- |
| Neutral | 75,374 |
| Happy | 134,915 |
| Sad | 25,959 |
| Surprise | 14,590 |
| Fear | 6,878 |
| Disgust | 4,303 |
| Anger | 25,382 |
| Contempt | 4,250 |
| None | 33,588 |
| Uncertain | 12,145 |
| Non-Face | 82,915 |
| Total | 420,299 |

Features provided by the AffectNet database:

* Images of the faces
* Location of the faces in the images
* Location of the 68 facial landmarks
* Eleven emotion and non-emotion categorical labels (Neutral, Happiness, Sadness, Surprise, Fear, Disgust, Anger, Contempt, None, Uncertain, No-Face)
* Valence and arousal values of the facial expressions in a continuous domain

**Proposed Work**

In this project, the main idea is to create a deep learning model to classify the faces images in terms of motions. In the past years, many researchers tried to develop their version of how emotions can be detected on the face. Besides the actual code that I am going to create, I will conduct some tests and see how my model performs in comparison with some other similar programs. The comparison will be related to the build of the program, meaning that I will describe the feature extraction process, classification, training and testing processes. A Deep Convolutional Neural Network (CNN) method will be used, which has an architecture that consists of filter layers and a classification layer. This approach has been chosen for its ability to deal with raw data and to overcome the disadvantage of feature-based methods that require a significant exertion should be put on to plan and utilise different include extraction techniques which are human-made features. CNN input and output are array vectors called a feature map. Since this project will deal with images, the array dimension will be 2D.

The code that is going to be build will use images with people showing the six universal emotions, which will be the dataset. First, the images from the dataset will be pre-processed. In order to process the pictures, a face detection code will be used. This will help the program to recognise the human face from the picture,

The second stage will be facial expression extraction. Basically, the program will have to recognise patterns that will indicate a specific emotion. The program will need to store just the optimal key points like eyes, eyebrows and mouth. After the extraction, the features are stored into a feature vector.

The next step would be the classification. The problem with the classifiers is that they assume that there is no correlation between features. The program will need to learn how to recognise the difference between the images. To do so, labels will be added, matching the six emotions.

Once all steps have been carefully completed, the training of the systems can begin. The dataset that will be used is AffectNet dataset which contains over one million facial images collected from the internet. This is by far the largest database of facial expression.

For the training bit, only half of the pictures will be used. The training process accuracy will depend on how correctly the system can classify the images so that the training process will proceed optimally. Moreover, to accelerate and reduce the cost of training, a pre-existing network model will be used. The idea behind transfer learning for image classification is that an effective generic model of the visual world can be obtained by training a model on a large and general enough dataset. You can then take advantage of these learned feature maps without having to start from scratch by training a large model on a large dataset. The only modification that needs to be made is to unfreeze a few of the top layers of a frozen model base and jointly train both the newly-added classifier layers and the last layers of the base model. This will allow to "fine-tune" the higher-order feature representations in order to make them more relevant to this task.

Finally, the last bit consists of testing. Once the training has been done, we can see how well the system can recognise the emotions from new pictures that were not used previously in the training process. The accuracy for each of the six emotions will be measured to see if there could be any modifications that can be done to improve the overall score so that the error percentage to be as small as possible.

Besides all the work done on the actual code, research will be carried out to identify the most suitable ways to develop a better understanding of what machine learning/deep learning is and how it works. Also, the language used for the code will be python which is a high-level program language. To master this type of language, extra time will be spent on reading and practising some exercises

**Aims**

* Develop a program using machine learning/deep learning that is capable to automatically detect and interpret the emotions expressed by a human face.
* Optimise the program in order to deliver accurate predictions.

**Objectives**

1. Carry out the literature review
   1. Research face detection technology
   2. Research facial emotion technology
2. Investigate suitable tools, techniques and methods to carry out the project
3. Finding the best suitable methods for building the program
   1. Find a face detection algorithm
   2. Find a feature extractor
   3. Find a classifier
4. Training the program to detect and recognise facial emotion from the pictures
5. Testing how accurate the trained program can detect facial emotions from new pictures
6. Conduct an analysis of how the system performed
7. Evaluate how the overall system works
   1. Discuss the results and highlight the good practice
   2. Discuss the flows of the system
8. Give Recommendations
   1. Conclude and recommend what I could have done differently and future work and research that can be done within the field.

**Skills**

|  |  |  |
| --- | --- | --- |
| Skills | Current skill description | Skills improvement via |
| Python programming | Skill level: low | Online lectures and practice sessions |
| Machine learning | Skill level: medium  Acquired via Machine Learning module during 3rd year | Continuing to follow the module and studying deeper my free time |
| Deep learning | Skill level: medium  Acquired via Machine Learning Module during 3rd year | Continuing to follow the module and studying deeper my free time |
| Data processing | Skill level: low | Utilisation of various Python libraries to facilitate the processing and analysis of datasets. |
| Time management | Skill level: medium  Acquired during other projects development |  |

**Resources**

Hardware:

* A suitable, modern and powerful CPU to facilitate the software requirements.
* A suitable, modern and powerful GPU to accelerate the machine learning training.
* Sufficient RAM memory to facilitate the loading of the dataset into program memory.

Software:

* Windows 10 as an operating system
* Machine learning/Deep Learning packages in python (Scikit-learn, tensorflow, keras,Numpy CUDA)
* Images dataset (AffectNet dataset)
* Anaconda3
* Jupyter notebook
* Pyton 3.8
* Github for cloud storage

**Report structure**

Abstract

1. Introduction
   1. Background presentation
   2. Aims
   3. Objectives
   4. Structure of the dissertation
2. Literature review
   1. General face detection
   2. General facial emotion detection
3. Machine Learning Techniques
   1. Datasets
   2. Feature selections
   3. Classifiers comparison
   4. Deep Learning exploration
4. Analysis
   1. Face detection
   2. Feature extraction
   3. Classification
   4. Training results
5. Evaluation, Results and Discussion
6. Conclusion and future work

**Marking scheme**

1. Project Type

Investigative Project

|  |  |
| --- | --- |
| Report | 90% |
| Viva | 10% |
| Total | 100% |

Report

|  |  |
| --- | --- |
| Abstract & Introduction | 5% |
| Analysis | 20% |
| Synthesis | 50% |
| Evaluation & Conclusion | 20% |
| Presentation | 5% |
| Total | 100% |

Viva

|  |  |
| --- | --- |
| Presentation | 50% |
| Discussion | 50% |
| Total | 100% |

1. Project report

|  |  |  |
| --- | --- | --- |
| Report | Section |  |
| Abstract & Introduction | 1.Abstract  2.Introduction | 5% |
| Analysis | 3.Literature review  4. Machine learning techniques | 20% |
| Synthesis: discussion of methods & results | 5.Description and justification of the approach to be taken  6.Description of the tools to be used  7.Testing and results discussion | 20% |
| Synthesis: quality of practical work | 8.Quality of deliverables  9.Demonstration of effective use of practical skill | 30% |
| Evaluation & Conclusion | 10.Conclusion and future work | 20% |
| Presentation | \* | 5% |
| Total | \* | 100% |

**Project plan**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Semester 1** | | | | | | | | | | | | | | | | | | | | | | | |
| Aim | Objectives | Tasks | Estimate Hour Required | | W1 | W2 | W3 | | W4 | | W5 | W6 | | W7 | | W8 | W9 | W10 | | | W11 | | W12 |
| Project Initiation | Project Initiation Document | PID Creation | 10 | |  | | | |  | |  |  | |  | |  |  |  | | |  | |  |
| PID Submission |  | |  |  | Submission | |  | |  |  | |  | |  |  |  | | |  | |  |
| Project Research Plan | Terms of Reference | Project Research | 15 | |  |  |  | | | | | | | | |  |  |  | | |  | |  |
| Project Planning | 15 | |  |  |  | |  | |  | | | | |  |  |  | | |  | |  |
| TOR Review | 5 | |  |  |  | |  | |  |  | |  | |  |  |  | | |  | |  |
| TOR Submission | \* | |  |  |  | |  | |  |  | |  | | Submission |  |  | | |  | |  |
| Investigating facial emotion detection algorithms and develop future knowledge | Research the principles of face detection and facial emotion detection | Chapter Planning | 2 | |  |  |  | |  | |  |  | |  | |  | |  | | |  | |  |
| Research face detection | 10 | |  |  |  | |  | |  |  | |  | |  | |  | | |  | |  |
| Research face emotion detection | 10 | |  |  |  | |  | |  |  | |  | |  |  | |  | |  | |  |
| Chapter write up | 6 | |  |  |  | |  | |  |  | |  | |  |  |  |  | |  | |  |
| Research various code components | Chapter planning | 2 | |  |  |  | |  | |  |  | |  | |  |  |  |  | |  | |  |
| Dataset reading and research | 10 | |  |  |  | |  | |  |  | |  | |  |  |  |  | | |  |  |
| Feature extraction reading and research | 10 | |  |  |  | |  | |  |  | |  | |  |  |  |  | | |  |  |
| Classification reading and research | 10 | |  |  |  | |  | |  |  | |  | |  |  |  | | |  |  | |
| Deep learning reading and research | 20 | |  |  |  | |  | |  |  | |  | |  |  |  | | |  |  | |
| Chapter write up | 6 | |  |  |  | |  | |  |  | |  | |  |  |  | | |  |  | |
| Analysis chapter submission | Analysis chapter submission | \* | |  |  |  | |  | |  |  | |  | |  |  |  | | |  |  | Submission |
| **Semester 2** | | | | | | | | | | | | | | | | | | | | | | | |
| Aim | Objectives | Tasks | Estimate Hour Required | W1 | | W2 | W3 | W4 | | W5 | W6 | | W7 | | W8 | | W9 | W10 | | W11 | | | W12 |
| Implementing all the data acquired from research by developing the detection engine and discuss the results | The creations of relevant algorithms for incoming data pre-processing | Chapter planning | 3 |  |  |  |  |  | |  |  | |  | |  | |  |  | |  | | |  |
| Pseudo-code for algorithms | 8 |  |  |  |  |  | |  |  | |  | |  | |  |  | |  | | |  |
| Code for algorithms | 10 |  | |  |  |  | |  |  | |  | |  | |  |  | |  | | |  |
| Chapter write up | 10 |  | |  |  |  | |  |  | |  | |  | |  |  | |  | | |  |
| Perform an analysis of a test of the detection engine on an unseen dataset | Chapter planning | 3 |  | |  |  |  |  |  |  | |  | |  | |  |  | |  | | |  |
| Testing methodology | 5 |  | |  |  |  |  |  |  | |  | |  | |  |  | |  | | |  |
| Product testing | 10 |  | |  |  |  | |  |  | |  | |  | |  |  | |  | | |  |
| Collecting testing result | 5 |  | |  |  |  | |  |  |  |  | |  | |  |  | |  | | |  |
| Chapter write up | 10 |  | |  |  |  | |  |  |  | | |  | |  |  | |  | | |  |
| Carry out an evaluation of the final detection engine and identify areas of improvement and future work | Chapter planning | 3 |  | |  |  |  | |  |  | |  | |  |  |  |  | |  | | |  |
| Results analysis | 8 |  | |  |  |  | |  |  | |  | |  |  |  |  | |  | | |  |
| Evaluation of results | 8 |  | |  |  |  | |  |  | |  | |  | |  |  | |  | | |  |
| Identification of areas of improvement | 8 |  | |  |  |  | |  |  | |  | |  | |  |  | |  | | |  |
| Chapter write up | 15 |  | |  |  |  | |  |  | |  | |  | |  |  | |  | | |  |
| Report and product submission |  |  |  | |  |  |  | |  |  | |  | |  | |  |  | | Submission | | |  |

# Appendix B: Project Initiation Document (PID)

**KV6003– Individual Project**

|  |  |  |  |
| --- | --- | --- | --- |
| Student Name | **Family Name**  **Neagu** | **Other names**  **Alexandru-Adonis** | **Usually known as**  **Alex** |

|  |
| --- |
| **Project Supervisor:**  **Dr. Ossama Alshabrawy** |
| **Aim of project:**  **This project aims to develop a program using machine learning/deep learning that is capable to automatically detect and interpret the emotions expressed by a human face.** |
| **Rationale for project:**  **Understanding the emotion of the people has a widespread consequence for society and business. A program like this could be implemented in airports to help detect terrorist activity or drug traffickers. Also, some companies are looking forward to using facial emotion recognition to evaluate the reaction of the people when they reveal a product predicting the level of customer satisfaction.** |
| **The main challenge is:**  **- Acquiring well-labelled training data.**  **- Identifying facial features**  **-Finding thre right pre-trained models**  **-Finding the right set of parameters for models** |
| **Type of product to be produced or investigative work to be undertaken:**  **The product is going to be a system that is using well trained data and feature identification to achieve highly accurate emotion detection on a human face.** |
| **Resources required:**  **- Machine learning/Deep Learning packages in python (Scikit-learn, tensorflow, keras)**  **- Imagine datasets (AffectNet dataset)**  **- Books**  **- Research papers**  **-Powerfull enough computer** |
| **Any external body involved? If so, who?**  **N/A** |
| **Signatures:**  A picture containing text  Description automatically generatedShape  Description automatically generated**Student: Supervisor:** |

# Appendix C: Import all the libraries required and configure the GPU for training

Text

Description automatically generated

# Appendix D: Import a pre-trained model from Keras using the weights from ImageNet

Text

Description automatically generated

# Appendix D: Use ImageDataGenerator() for data augmentation

Text

Description automatically generated

# Appendix E: First configuration stage

Text

Description automatically generated

# Appendix F: Second configuration stage

Text

Description automatically generated

# Appendix G: Third configuration stage-Adagrad

Text

Description automatically generated

# Appendix H: Third configuration stage-Adadelta

Graphical user interface, text

Description automatically generated

# Appendix I: Fourth configuration stage-learning rate 0.5

Text

Description automatically generated

# Appendix J: Fourth configuration stage-learning rate 0.01

Text

Description automatically generated

# Appendix J: Fifth configuration stage

Text

Description automatically generated

# Appendix K: Sixth configuration stage

Text, letter

Description automatically generated

# Appendix L: Seventh configuration stage

Text

Description automatically generated

# Appendix M: Eighth configuration stage

Text

Description automatically generated

# Appendix N: initialize the tuner to find the best model

Text

Description automatically generated

# Appendix O: initialize the tuner to find the best model

A picture containing text, orange

Description automatically generated

# Appendix P: Train the model and save the best weights using ModelCheckpoint

Text

Description automatically generated

# Appendix Q: Plot the accuracy and cross-entropy diagrams

Text

Description automatically generated

# Appendix R: Import all the libraries required and configure the GPU for training

Text

Description automatically generated

# Appendix S: Use ImageDataGenerator() for data augmentation in testing

Graphical user interface, text

Description automatically generated

# Appendix T: Create a list with the label of each picture

Text

Description automatically generated

# Appendix U: Create a result map

Text

Description automatically generated

# Appendix V: Make prediction on the validation set and create a list with the name of the emotion predicted

A picture containing text

Description automatically generated

# Appendix W: Create and plot confusion matrix

Text

Description automatically generated

# Appendix X: Create and print the classification report



# Appendix Y: Create and plot the ROC curve for each class

Text

Description automatically generated

# Appendix Z: Create and plot the PR curve for each class

Text

Description automatically generated