Automated Facial Expression Detection and Emotion Recognition Using Chained Neural Networks in Python

Afonso B. Caniço
ambco@iscte-iul.pt
Instituto Universitário de Lisboa (ISCTE-IUL)
Portugal

Samuel R. Correia
smrca2@iscte-iul.pt
Instituto Universitário de Lisboa (ISCTE-IUL)
Portugal

Abstract

The introduction of novel Neural Network (NN) architectures such as Convolutional Neural Networks (CNN) has allowed the broadening of the Machine Learning (ML) field to now-standard fields such as Computer Vision. In this regard, research has been conducted and implemented into applications such as object detection and image classification. In this paper, the utilization of such algorithms to establish a system for the detection of people's faces and the automatic determination of their emotions based on their facial expressions in a given image or video is explored.

CCS Concepts: • **Computing methodologies** → **Object detection**; **Supervised learning by classification**.

Keywords: computer vision, emotion recognition, convolutional neural networks

Reference Format:
Afonso B. Caniço and Samuel R. Correia. 2023. Automated Facial Expression Detection and Emotion Recognition Using Chained Neural Networks in Python. In *Deep Learning for Computer Vision (DLCV), 2023, Iscte - Instituto Universitário de Lisboa.* 8 pages.

1 Introduction

This project aims to explore the usage of Convolutional Neural Networks in Computer Vision by constructing a neural network architecture capable of detecting human faces in pictures and automatically classifying their portrayed emotions. The primary objective is to study CNNs and their potential applications in real-world fields. The emphasis lies not only on employing neural networks for automatic object detection in images but also on their automatic classification.

This paper details the methodology and tools used in implementing the proposed architecture in Python and additionally provides an analysis of the produced results and possible shortcomings.

This project was developed as part of an elective Deep Learning for Computer Vision (2023/24) course as part of Iscte-IUL's[1] Master's in Computer Engineering programme.

2 Related Work

2.1 Face Detection

Much of the existing literature on face detection uses this detection process as the first step in analysing an individual's face, which is also the case for this project. While research and notable results in this field date back to 1991 with the usage of mathematical concepts such as eigenvectors [15] or active shape models (ASM) [17], modern solutions usually rely on the usage of neural networks (NN) to achieve fast, reliable, and robust results [6]. It should be noted that, even though most published research usually focuses on the usage of these NNs to create face detection systems, other methods have certain advantages which may provide superior performance for some specific tasks.

2.2 Facial Expression Emotion Classification

A review of the published literature reveals most datasets on emotion classification focus on 6 or 7 emotion categories (e.g. [9], [16], [12]), with markedly fewer datasets (e.g. [11], [4]) introducing more nuanced emotions or characteristics into their classification [10].

As for the classification models, a recent survey has concluded that current state-of-the-art (SOTA) facial expression emotion classification relies on CNNs and adjacent NN architectures to achieve accurate results [13]. Noteworthy among these is the work by [14], which employs ResNet [7]. Due to the subjectivity of determining emotions, especially when classifying with 7 different categories, expected accuracies for these models are lower than initially expected. Ideal accuracies for these types of problems are at about 80% while the baseline lies at about 50% [14].

3 Approach

A two-fold approach is employed to assess the emotions conveyed by facial expressions in a given image. Firstly, the regions of the images containing faces are detected, and

[1] https://www.iscte-iul.pt/

Table 1. Test results for the trained YOLOv8 model.

TP	FP	FN
301	41	9

secondly, these faces are analysed to determine the emotions they portray.

As the primary objective is to initially identify the regions of the image containing faces and subsequently analyze them, the focus is on obtaining bounding boxes outlining the faces. To achieve this, a model is trained to derive the coordinates of these bounding boxes using a dataset consisting of numerous images annotated with face bounding-box coordinates [1].

After obtaining the image fragments corresponding to only the faces detected in the original images, a second model is applied to determine which emotion from a previously defined collection is most likely to correspond to the face being analysed. This step also requires the creation and training of a model, which in this case is done using a dataset comprised of images of facial expressions annotated with their corresponding emotions. Two different datasets will be used, as well as three distinct CNN architectures, results obtained from these different approaches will be compared. Specifically, the FER+ [3] and AffectNet [12] datasets will be used to train a custom CNN model built "from scratch" as well as two models that utilise transfer learning.

With both modules working in tandem, a system which can receive any given image and determine what emotion the faces portrayed in it are presenting will be created. Additionally, this system shall be capable of analysing video to dynamically determine emotions.

4 Implementation and Results

This approach is implemented as PyFER[2], a Python library for automated facial expression detection and emotion recognition built using the TensorFlow[3] [2], Ultralytics[4], and OpenCV-Python[5] libraries.

4.1 Face Detection

The face detection system created is based on the YOLOv8 object detection model [8]. This model was trained with the dataset previously specified to generate bounding boxes around each of the faces of all individuals on a given image. The results obtained from the creation of this model were more than satisfactory, as a very reliable model was obtained. In particular, the confusion matrix showcased in table 1 shows that the model achieved an accuracy value of approximately 86% and an F1-Score of approximately 92%.

The Precision-Recall and F1-Confidence curves for the generated model are presented in figures 1 and 2.

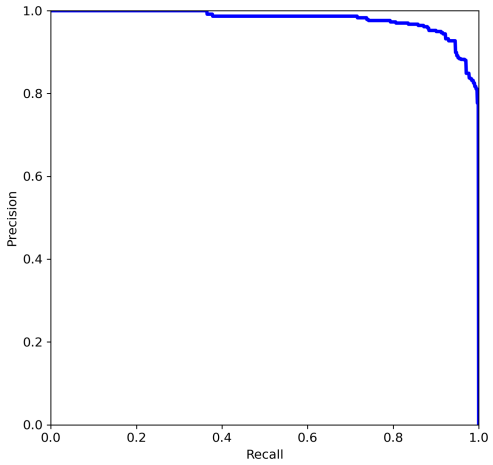

Figure 1. PR curve for the trained YOLOv8 model

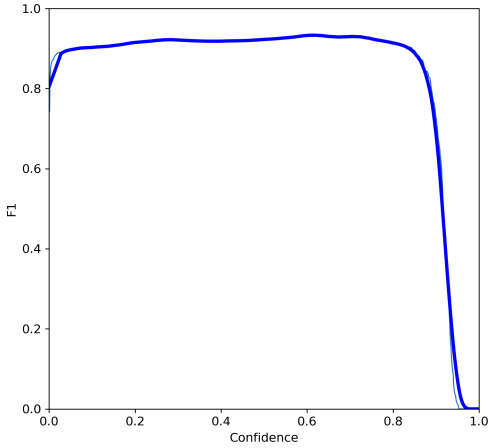

Figure 2. F1 curve for the trained YOLOv8 model

4.2 Facial Expression Classification

Creating a facial expression classification model with satisfactory results was more challenging than initially expected. 2 different datasets were used to train a total of 6 distinct models, and results were compared between them to obtain the best possible model.

3 different models were trained with each dataset. These 3 models consist of a CNN employing transfer learning from ResNet50V2[6], another transfer learning based model with MobileNetV2[7], as well as a fully custom CNN built "from scratch". All the models have the same basic structure:

[2] https://github.com/ambco-iscte/pyfer
[3] https://www.tensorflow.org/
[4] https://www.ultralytics.com/
[5] https://github.com/opencv/opencv-python
[6] https://www.tensorflow.org/api_docs/python/tf/keras/applications/resnet_v2/ResNet50V2
[7] https://www.tensorflow.org/api_docs/python/tf/keras/applications/mobilenet_v2/MobileNetV2

- Data augmentation layers, to avoid overfitting;
- A convolutional model for feature extraction. Note that the pre-trained models used for the transfer learning models had their 6 last layers unlocked for training;
- Finally, the classifier, consisting of several dense neural network layers.

The exact architecture that was created for these models is described in Appendix A.

4.2.1 FER. To train these, first the original FER dataset [5] was used. This dataset contains 28,709 48x48 greyscale images of faces depicting one of seven emotions. It is important to note that FER is an unbalanced dataset, with only a subset of the emotion labels (in particular, Neutral, Happiness, Surprise, Sadness, and Anger) accounting for over 90% of the images.

The initial results obtained from these models were not ideal, with the best results being achieved by the custom CNN. This model achieved an accuracy of approximately 65% with a loss value of 0.93.

It was hypothesised that the sub-par results of the transfer learning model, even with its final layers unlocked for fine-tuning, could be attributed to the low image resolution presented in the utilised dataset, which clashes with the higher-resolution images used to train the ResNetV2 and MobileNetV2 models.

It was also brought to attention that relying solely on accuracy and loss metrics for the evaluation strategy was less than ideal, particularly when dealing with unbalanced datasets such as the FER dataset. As such, for the evaluation of models, precision and recall were also considered. These metrics were also considered when selecting the epoch where the model presented the best performance.

Additionally, attempts were made to combat some problems which had been encountered due to the unbalance of the dataset. To do this, class weights were employed, ensuring that highly common classes, such as "Neutral," are not over-represented, and vice versa.

When looking at some of the images in the FER dataset and their respective labels, it is clear that not all are classified accurately. To remedy this, labels from the FER+ dataset[8] were used. This dataset attempts to create more accurate labels by using the labels assigned by 10 independent taggers, for the training of the model the "correct" class was defined to be the mode of these tags. This dataset also includes two additional labels not included in the original FER dataset: Unknown and No Face (NF). Images classified as one of these two classes were discarded as they represent a very small portion of the dataset and won't be very relevant in the context of PyFER.

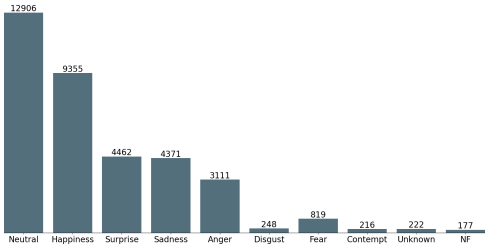

Figure 3. Emotion label distribution for the FER+ dataset

Additionally, TensorFlow was configured to take advantage of the system's graphics card. This is reflected in the total runtime of training the models being reduced to less than one-fifth of the original training time.

Table 2. Metrics obtained from each model trained with FER+

Model	Loss	Accuracy	Precision	Recall
Custom CNN	0.79	0.71	0.79	0.62
TL MobileNetV2	1.51	0.42	0.64	0.12
TL ResNet50V2	1.44	0.44	0.68	0.17

After these changes, the obtained models performed better on all metrics and their training time was reduced significantly.

While the models utilizing transfer learning also exhibited improvement, the Custom CNN model still outperforms these, having an accuracy of approximately 71% and 0.79 loss. Notably, both models that used pre-trained CNNs demonstrated very similar performance. They also presented a low recall score, indicating that the ratio of correctly identified images associated with any specific expressions to all actual instances of those expressions in the dataset is relatively low.

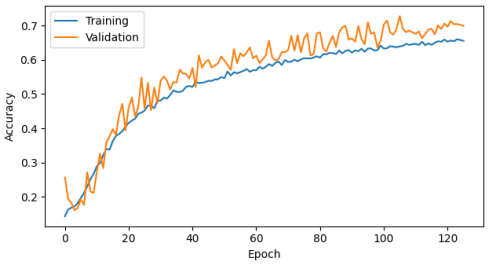

Figure 4. Accuracy for the Custom CNN trained with FER+

[8] https://github.com/Microsoft/FERPlus

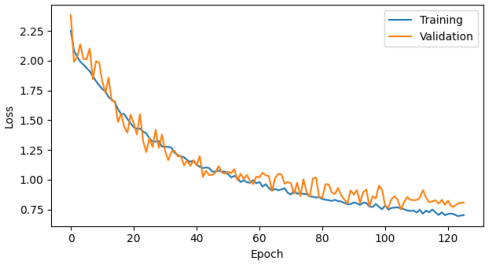

Figure 5. Loss for the Custom CNN trained with FER+

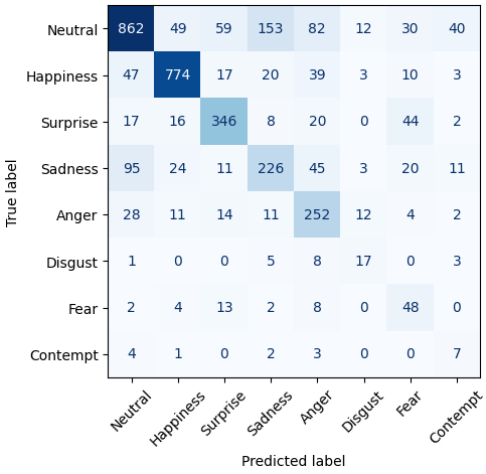

Figure 6. Confusion Matrix for the Custom CNN trained with FER+

After all the improvements made, the custom CNN model trained with the FER+ dataset presented very satisfactory results, while needing relatively little training time. This model can, quite reliably, correctly identify facial expressions. (see figures 4, 5, 6).

4.2.2 AffectNet. The truncated version of the AffectNet dataset was employed, containing approximately 290 thousand (290,000) images, as the full dataset contains over 140 gigabytes of data, falling well outside the scope of this project. The truncated dataset is composed of images of faces cropped to 224×224 pixels, with most images being in colour. These images are labelled with one of 10 emotions, and, as with the FER dataset, their distribution throughout these emotions is not balanced. From these 10 emotions, 3 were discarded: None, Uncertain, and Non-Face, as once again these represent a small portion of the dataset and won't be of interest to the model. Class weights were used once more to lessen the impact of the dataset's imbalance.

It was expected that, since these images more closely reflect the dataset that was used to train the ResNet model, the transfer learning model's results would improve and possibly even surpass the results obtained with the custom CNN.

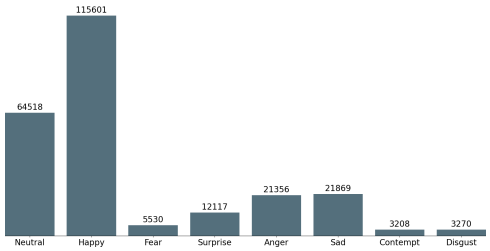

Figure 7. Emotion label distribution for the AffectNet dataset

This dataset, just like the previous one, required some pre-processing to create separate training, validation, and test subsets.

It was immediately clear that, due to the large size of this dataset, model training would take significantly longer than with the FER+ dataset. To be able to present results in a reasonable timeframe, the model was restricted to use a sample of fifty thousand (50,000) images, downscaled to 128×128 pixels. Even so, the time required to train the models was very lengthy.

In the case of the custom CNN model, training led to the system shutting down multiple times during training even after putting these restrictions in place, possibly due to excessive continued processing load. These hardware limitations were resolved using a more controlled setup[9]. The training of the custom CNN with AffectNet took over 1 day and 5 hours to finish.

Table 3. Metrics obtained from each model trained with AffectNet

Model	Loss	Accuracy	Precision	Recall
Custom CNN	1.04	0.59	0.71	0.44
TL MobileNetV2	1.45	0.45	0.77	0.20
TL ResNet50V2	1.40	0.49	0.79	0.22

However, analysing the results of the trained models revealed that the evaluation metrics for the best-performing model: Custom CNN were outperformed by the model trained using the FER+ dataset. The custom AffectNet-trained model was only able to achieve accuracy and loss values of approximately 59% and 1.04, respectively. These results are showcased in figures 8, 9, and 10.

[9]The laptop used to train the models was placed on top of an external laptop cooler and had a standing fan pointed towards it 24/7 during this process. Surprisingly, this worked.

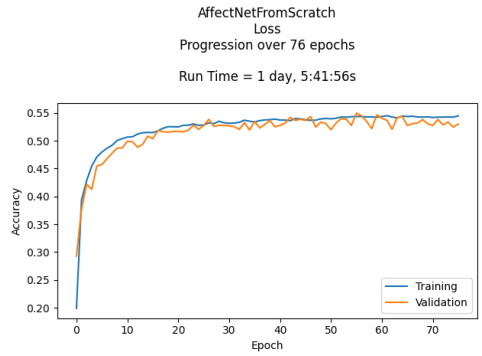

Figure 8. Accuracy for the Custom CNN trained with AffectNet

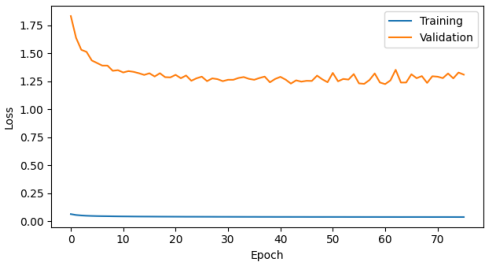

Figure 9. Loss for the Custom CNN trained with AffectNet

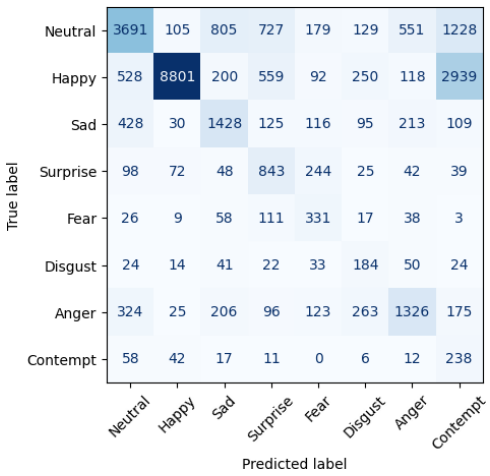

Figure 10. Confusion Matrix for the Custom CNN trained with AffectNet

When comparing the results obtained from both datasets it is important to consider that the metrics obtained are from different test sets. As such, one must not only compare these metrics but also consider the broader context given by the images in both datasets.

When looking at figure 9, on the loss observed per epoch, it is immediately apparent that the loss for the training set is, from the start, exceedingly low while the values for the validation sets are, expectedly, quite high. This occurred exclusively with all models trained with the AffectNet, even when trained using the same settings as with the FER+ dataset. Other methods of calculating loss also revealed the same results. The differences between the datasets are not believed to explain this intriguing problem, as they are both fairly similar and also contain similar labels to classify the images.

Observing good scores for training data and relatively bad ones for validation data is usually an indicator of over-fitting, however, when considering the other metrics, it is possible to determine that this is not the case as, for instance, the accuracy for the training and validation sets is very similar. The exact cause of this issue remains undetermined. Failing to determine any reasonable reason for this to occur, it is assumed that it is due to a fault with the method used to calculate the loss metric.

As with the FER+ dataset, the best results were obtained with the Custom CNN, it did however take close to 3 times as long to train than either pre-trained model, requiring close to 35 hours to complete its training.

It was satisfying to observe that, as anticipated, the transfer-learning models performed better with this dataset than the previous one, even if only marginally. This is attributed to the fact that the images of this dataset are, in contrast to the previous, in colour and of a relatively high resolution, making them more similar to the data that was used to train these models originally. The second prediction made was, in fact, not correct as these models were not able to outperform the Custom CNN model.

However, the custom CNN model did present slightly worse metrics when trained with this dataset. This can be an indicator that the labels of the images of the AffectNet dataset are not quite as precise, or that the intensity of the emotions presented in the images is not as high as in FER+. From just looking at some of the images in the dataset it is not possible to discern any specific factor which can differentiate the two datasets, besides the aforementioned characteristics of image colour and dimension.

While it is quite certain that the custom models are preferable for usage in PyFER, the choice between which of the two models presents better real-world results is still open.

5 PyFER

After designing and training the models for the two essential parts of this system, a complete pipeline was created which, given an image, can detect any faces present in said picture and determine which emotion each face is most likely presenting.

Figure 11. Simple PyFER Pipeline

Simply put, when applying PyFER to an image, it first passes it through the trained YOLOv8 face detection model from section 4.1. This model returns the coordinates which define a box around each of the detected faces, which are then sent to one of the emotion classification models specified in Section 4.2. The emotion classification model then returns the emotion which it deems to most likely correspond to the face provided as well as its confidence in said prediction. Finally, the user is presented with the original image overlayed with the boxes outlining the faces detected, where each face-outlining box is labelled with the predicted emotion and prediction confidence.

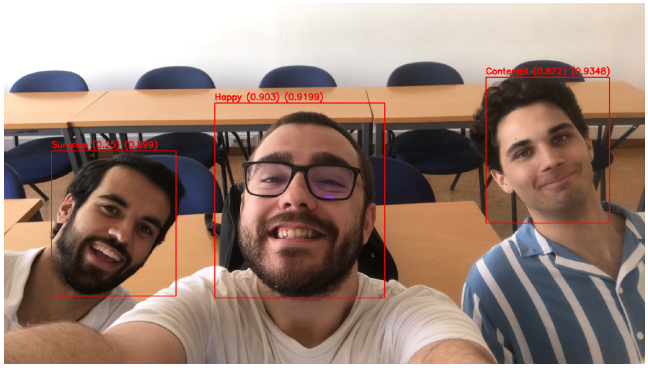

Figure 12. Example of PyFER output when applied to an image.

PyFER is also capable of processing a video stream from, for example, a webcam, and automatically applying the process previously described, outputting a video stream (with a slightly lower frame rate) with labelled boxes around each detected face.

The creation of PyFER was, on occasion, stalled due to issues related to the versions of the software that was used as well as the operating system.

For instance, to allow TensorFlow to use the graphics card to enhance performance on a Windows system, it is necessary to run the software on a Linux virtual machine[10] due to compatibility issues. While this was sufficient for training the model, this virtual machine does not have access to USB accessories, such as webcams, and thus, such video streams could not be captured for application in PyFER. To solve this issue, PyFER has to either run on native Windows without using the GPU, or several configuration steps must be followed, which involve manually rebuilding the WSL kernel to allow for USB accessory access. The system was configured for both of these options. However, running PyFER in Windows without the usage of the GPU seems to be the most favourable option, as the performance benefit is not nearly as noticeable as during the training phase, and the setup is much more simplified. This aligns with the goal of providing PyFER as a ready-to-use and easily configurable library.

One other problem that was faced involved the format used to save the models. The trained models were saved with the recommended *.keras* format. This format is, however, very sensitive to the TensorFlow version used, to a point where if any other version of TensorFlow is used to open these models (be it older or newer) the models would fail to load. Thus, the models could only be loaded when using the precise versions of the software used during their training. The models were thus re-trained and saved in TensorFlow's legacy *SavedModel* format, which seems to be portable when it comes to TensorFlow version mismatches.

After resolving these issues, a robust tool was created which can take as input any given image and label it with the locations of the faces in these images as well as label those faces with the most likely portrayed emotions. PyFER's performance additionally allows users to automatically evaluate video streams with a frame-by-frame analysis, outputting a video stream with, in most cases, a similar frame rate.

6 Discussion

For the creation of PyFER, an automated facial expression detection and emotion recognition program, two essential challenges were faced: detecting faces in images, and determining the emotion portrayed by each face. For each challenge, a system using convolutional neural networks was created to obtain fast and robust results.

The face detection system is based on the YOLOv8 object detection model. This model was trained with a dataset of images with annotations defining the bounding boxes around each face. This model was relatively easy to create and provided very good results.

For the facial expression classification model, 3 CNN architectures (1 "from scratch" model, and 2 created with transfer learning) were designed, which were then trained with two distinct datasets (FER+ and AffectNet). The models which presented the best results, which also took the longest to train, were the ones using the custom CNN architecture. These two models present very satisfactory results, and either of the two can be used to great effect in PyFER.

Finally, the creation of PyFER consisted of creating a pipeline which would, in a sense, connect these systems to produce the labelled images. The performance of this system also enabled PyFER to be used with, for instance, a webcam, to automatically label frames in a video stream from a device such as a webcam with a frame rate comparable to the original frame rate.

[10]Windows Subsystem for Linux version 2 (WSL2) was used.

During the creation of PyFER and its main systems, several challenges were faced, including some very complex problems stemming from the software that was chosen to be used. These issues were almost all resolved. The only notable limitation that remains is that it is not possible to use a graphics processor's acceleration when analysing a webcam's video stream without following several very complicated steps to create a custom virtual machine, as TensorFlow does not support native GPU access on Windows. Nevertheless, the performance increase observed when using a graphics card in PyFER is considered to be negligible, as the biggest performance bottleneck seems to be related to training the models.

Finally, the following questions are proposed which are believed to warrant future work:

- Is it possible to create a test to determine which of the emotion classification models present the best real-world performance?
- Are there any other emotion classification models which could provide good results?

Acknowledgments

We kindly thank Dr. Mohammad H. Mahoor, Professor of Electrical and Computer Engineering at the University of Denver, and M. Mehdi Hosseini, Ph.D. Student of Electrical and Computer Engineering at the University of Denver, for providing us with the AffectNet dataset to aid in the development of our facial expression classification model.

We kindly thank Dr. Jeffrey Cohn and Megan Ritter from the University of Pittsburgh for providing us with the Cohn-Kanade dataset and its extended version to aid in the development of our facial expression classification model. While we ended up not utilizing this dataset to train our model, we appreciate being provided with it.

References

[1] 2023. Human Faces (Object Detection). https://www.kaggle.com/datasets/sbaghbidi/human-faces-object-detection

[2] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dandelion Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. 2015. TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems. https://www.tensorflow.org/ Software available from tensorflow.org.

[3] Emad Barsoum, Cha Zhang, Cristian Canton Ferrer, and Zhengyou Zhang. 2016. Training Deep Networks for Facial Expression Recognition with Crowd-Sourced Label Distribution. In *ACM International Conference on Multimodal Interaction (ICMI)*.

[4] C. Fabian Benitez-Quiroz, Ramprakash Srinivasan, and Aleix M. Martinez. 2016. EmotioNet: An Accurate, Real-Time Algorithm for the Automatic Annotation of a Million Facial Expressions in the Wild. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 5562–5570. https://doi.org/10.1109/CVPR.2016.600

[5] Dumitru, Ian Goodfellow, Will Cukierski, and Yoshua Bengio. 2013. Challenges in Representation Learning: Facial Expression Recognition Challenge | Kaggle. https://www.kaggle.com/c/challenges-in-representation-learning-facial-expression-recognition-challenge/overview

[6] Md Khaled Hasan, Md. Shamim Ahsan, Abdullah-Al-Mamun, S. H. Shah Newaz, and Gyu Myoung Lee. 2021. Human Face Detection Techniques: A Comprehensive Review and Future Research Directions. *Electronics* 10, 19 (2021). https://doi.org/10.3390/electronics10192354

[7] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2015. Deep Residual Learning for Image Recognition. *CoRR* abs/1512.03385 (2015). arXiv:1512.03385 http://arxiv.org/abs/1512.03385

[8] Glenn Jocher, Ayush Chaurasia, and Jing Qiu. 2023. *YOLO by Ultralytics*. https://github.com/ultralytics/ultralytics

[9] Takeo Kanade, Jeffrey Cohn, and Yingli Tian. 1970. Comprehensive Database for Facial Expression Analysis. *4th IEEE International Conference on Automatic Face and Gesture Recognition, France* (02 1970).

[10] Shan Li and Weihong Deng. 2018. Deep Facial Expression Recognition: A Survey. *IEEE Transactions on Affective Computing* PP (04 2018). https://doi.org/10.1109/TAFFC.2020.2981446

[11] Shan Li and Weihong Deng. 2019. Reliable Crowdsourcing and Deep Locality-Preserving Learning for Unconstrained Facial Expression Recognition. *IEEE Transactions on Image Processing* 28, 1 (2019), 356–370.

[12] Ali Mollahosseini, Behzad Hasani, and Mohammad H. Mahoor. 2019. AffectNet: A Database for Facial Expression, Valence, and Arousal Computing in the Wild. *IEEE Transactions on Affective Computing* 10, 1 (2019), 18–31. https://doi.org/10.1109/TAFFC.2017.2740923

[13] Muhammad Sajjad, Fath U Min Ullah, Mohib Ullah, Georgia Christodoulou, Faouzi Alaya Cheikh, Mohammad Hijji, Khan Muhammad, and Joel J.P.C. Rodrigues. 2023. A comprehensive survey on deep facial expression recognition: challenges, applications, and future guidelines. *Alexandria Engineering Journal* 68 (2023), 817–840. https://doi.org/10.1016/j.aej.2023.01.017

[14] Lianzhi Tan, Kaipeng Zhang, Kai Wang, Xiaoxing Zeng, Xiaojiang Peng, and Yu Qiao. 2017. Group emotion recognition with individual facial emotion CNNs and global image based CNNs. 549–552. https://doi.org/10.1145/3136755.3143008

[15] Matthew A. Turk and Alex P. Pentland. 1991. Face recognition using eigenfaces. 586 – 591. https://www.scopus.com/inward/record.uri?eid=2-s2.0-0026384289&partnerID=40&md5=93093b2ec435893c01c0181657656ba4 Cited by: 4472.

[16] Michel Valstar and M. Pantic. 2010. Induced disgust, happiness and surprise: An addition to the mmi facial expression database. *Proc. Int'l Conf. Language Resources and Evaluation, Workshop EMOTION* (01 2010), 65–70.

[17] Shuicheng Yan, Ce Liu, Stan Z. Li, Hongjiang Zhang, Heung-Yeung Shum, and Qiansheng Cheng. 2003. Face alignment using texture-constrained active shape models. *Image and Vision Computing* 21, 1 (2003), 69 – 75. https://doi.org/10.1016/S0262-8856(02)00136-1 Cited by: 49.

A Facial Expression Recognition - CNN Architecture

A.1 Base Architecture

```
RandomFlip("horizontal", input_shape=(img_height,
    img_width, 3)),
RandomRotation(0.2),
Rescaling(1. / 255),
```

```
# The CNN to be used, either the pre-trained model
     or the custom CNN
convolutional_model,

Dropout(0.5),
Flatten(),
Dense(128),
BatchNormalization(),
Activation('relu'),
Dropout(0.2),
Dense(128),
BatchNormalization(),
Activation('relu'),
Dropout(0.2),
Dense(num_classes, activation='softmax')
```

A.2 Custom CNN

```
Conv2D(
    filters=64,
    kernel_size=(5, 5),
    input_shape=(img_width, img_height, 3),
    activation='elu',
    padding='same',
    kernel_initializer='he_normal',
),
BatchNormalization(),
Conv2D(
    filters=64,
    kernel_size=(5, 5),
    activation='elu',
    padding='same',
    kernel_initializer='he_normal',
),
BatchNormalization(),
MaxPooling2D(pool_size=(2, 2)),
Dropout(0.4),
Conv2D(
    filters=128,
    kernel_size=(3, 3),
    activation='elu',
    padding='same',
    kernel_initializer='he_normal',
),
BatchNormalization(),
Conv2D(
    filters=128,
    kernel_size=(3, 3),
    activation='elu',
    padding='same',
    kernel_initializer='he_normal',
),
BatchNormalization(),
MaxPooling2D(pool_size=(2, 2)),
Dropout(0.4),
Conv2D(
    filters=256,
    kernel_size=(3, 3),
    activation='elu',
    padding='same',
    kernel_initializer='he_normal',
),
BatchNormalization(),
Conv2D(
    filters=256,
    kernel_size=(3, 3),
    activation='elu',
    padding='same',
    kernel_initializer='he_normal',
),
BatchNormalization(),
MaxPooling2D(pool_size=(2, 2))
```