Facial Expression Recognition of emotion with Deep Learning CNN

Gabriele Cotigliani[1]

Other group members:
Ahnaf A. Majid[2], **Vipin Kumar**[3], **Laura Dragomir**[4]

Abstract. In this approach to Deep Learning, I developed a Convolutional neural network (CNN) for facial expression recognition (FER). I classified the expressions by emotions, according to the 6+1 most common, universal facial key emotions (the 1 will be our neutral class): 0 = anger,1 = disgust,2 = fear,3 = happiness, 4 = sadness, 5 = surprise, 6 = neutral. The dataset used for the training is FER2013 containing 35887 images of facial expression examples in grayscale. To build my model I used Keras, a high-level Neural Network library, applied as a wrapper on Tensorflow. The latter is an open-source library for machine learning, having both high and low-level APIs. Tensorflow is used as a backend for the model. I used both Google Colab, which is an online environment that offers the use of CPU, GPU or TPU to train models and Anaconda, using PyCharm as IDE installing CUDA and cudNN from Nvidia (for the use of GPU), to test the efficiency on a local machine and the difference. At first, I had to face Overfitting (the model overfits on the data training details such as noise rather than the general features that we are looking for), which I faced with restructuring the model (reducing the complexity), adding regularisation method Dropout to the model and via data augmentation. The overall result: I got an accuracy of 91% and a validation accuracy of 66%. My goals were to maximise accuracy with the resources I had and apply the result to images of different contexts and features.

Keywords: Convolutional Neural Network, Artificial Intelligence, FER, Emotions, Deep Learning

1 Introduction

The universality of facial expressions of emotions One of the most protracted debates in biological and social science. The universality of facial expressions of emotions among human beings from different cultures and backgrounds. Although many studies confirm this theory, on the other hand, many others do not believe in such a universality. In this paper, [6] the study tries to prove this by comparing Western with East Asians expressions, underlining the difference in non-verbal communication and facial expressions. For example, East Asians (non-Western culture influenced) tend to express non-verbal communication more with their eyes region, whether with their mouth area (such us) as explained better in this article from Nature [5] where they use the first paper as back-end. Although those are high-level concepts, we can agree that certain expressions are not always equal to the suggested emotion, especially across different cultures. However, we are not going to dive in too deep. For this research, we will "believe" in what Darwin started as a theory, when he first expressed his view about how emotions are expressed and manifested among humans and animals in similar ways [1] and many others pursuit it during the years, such as Silvan Tomkins and Paul Ekman. The latter, claiming the universality of facial expression of emotions of those 6, across different cultures. [3]

"Emotions change how we see the world and how we interpret the actions of others. We do not seek to challenge why we are feeling a particular emotion; instead, we seek to confirm it." - Ekman, P. (2003).[3]

However, we will see that our model is somehow in part impacted by these non-universal expressions, being trained by a dataset such as FER2013 which is unbalanced, e.g., the largest class, happy, containing more data 8989 (images), against the smallest class, disgust, with only 547 images—also, having a large amount of noise and non always apparent features. We will notice some incorrect predictions once performed on an external image, with a non-obvious expression, but a human can easily comprehend. Also, these images are a random collection, sometimes badly classified as some images are challenging to be recognised even by human perception, especially in the happy class, some may appear more like a surprise, and same for anger as fear. The reason why I carried on using this dataset is that being the first time training a model, grasping the beauty and complexity of Deep Learning, I wanted to start somewhere at the beginning, so I challenged myself to achieve a good result and understanding over such data and then, move on. Also, although it seems, it is not an easy dataset to overcome and achieve accurate validation accuracy, the reason is quality and noise over the images.

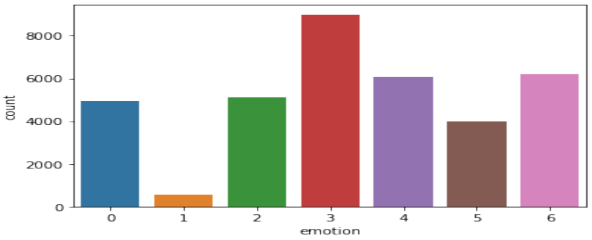

Figure 1. Emotions Graph

[1] School of Computing and Mathematical Sciences, University of Greenwich, London SE10 9LS, UK, email: gc9561r@gre.ac.uk

[2] School of Computing and Mathematical Sciences, University of Greenwich, London SE10 9LS, UK, email: am1513d@gre.ac.uk

[3] School of Computing and Mathematical Sciences, University of Greenwich, London SE10 9LS, UK, email: vk1104z@gre.ac.uk

[4] School of Computing and Mathematical Sciences, University of Greenwich, London SE10 9LS, UK, email: ld8431v@gre.ac.uk

2 Background

Why would we need such a feature (FER)? I will briefly describe some Real-life applications for this feature. A deep explained scenario can be the following: Children and adults who have ASD (autism spectrum disorder) have difficulty recognising those expressions; hence they genuinely struggle with basic communication. Last year I approached a possible solution, researching the feasibility of Augmented Reality in real-world applications. In particular, we saw important results in the children, proved by many studies, [2] [10] helping them recognise the expression via AR visualisations of results on an HMD (head-mounted devices), notably the Google Glass. In this example, computer vision elaborates in real-time the person's expression, thus, his emotion, via face recognition and FER (Face expression recognition) and predicts it with the help of a trained Neural Network, displaying the visual prediction on the HDM display. As described by this article [14], we can summarise other real-world examples for the application of FER. Education: observing students' engagement and focus, monitoring over new educational tools; Human-Computer Interaction and Usability: facial expression can provide valuable information about user experience, and the ease of use efficiency of the interface used; Security and Safety: driver fatigue detection, smart border control (airport), shoplifters detection, lie-detecting (crime and forensics), public place monitoring predicting possible treats (anti-terrorism); Healthcare: monitoring a patient's treatment, or perceive depression in an individual, Psychology: how do people respond to particular stimuli; Consumer reaction analysis and advertising: how do their react over a new product unboxed, or how do people react to new commercials or products;

In this paper, I will present my approach based on Convolutional Neural Network (CNN) for Facial Expression recognition of emotions. As input, it will take images, and once trained; it will be able to perform prediction of facial expression as emotion on those images, not necessarily from the same dataset. The dataset used to train and test our model is FER2013, designed by Goodfellow[7], structured in 7 folders (labels) for each emotion, respectively: anger, disgust, fear, happiness, neutral, sadness, surprise. Piece together by Google search results, of each emotion and similar. A CNN is one of the best approaches to perform imaging classification over many classes.

3 Experiments and results

Convolutional Neural Network The Idea behind a Convolutional Neural Network (CNN), in summary, is that we are structuring an artificial model as our brain so that it will think and see, thus processing data just like we do. We naturally predict everything we see, based on what we learned in the past, on the things we labelled and patterns we recognise. So after a CNN has been fed with lots of data, e.g., images, it will start to recognise general features and patterns between these pixel values (numbers), from detecting edges at first to more sophisticated shapes later, finding similarity and predicting, i.e. labelling it. A ConvNet (CNN) arranges his neurons from the start until the classification, in three dimensions: width, height and depth. In our case, the depth would be the colours, but we will train using grayscale images, hence depth = 1. The first part is called Hidden Layers, where the Network performs convolutions to "extract", i.e., filter those features. A convolution happens by applying the kernel (filter), which we can imagine as a matrix, in our case 3x3, over the image and performing a matrix multiplication between these two within a given point, which will generate a third value called feature map. When executed, the filter will slide over the input, performing the matrix multiplication at that given point and summing the result, which will then be stored into the corresponding coordinates in the feature map. I find this very fascinating. The second part is called Classification, where the ConvNet uses fully connected layers to classify these obtained features assigning probability based on what is the object predicted to be. However, enough theory, let us see my ConvNet.

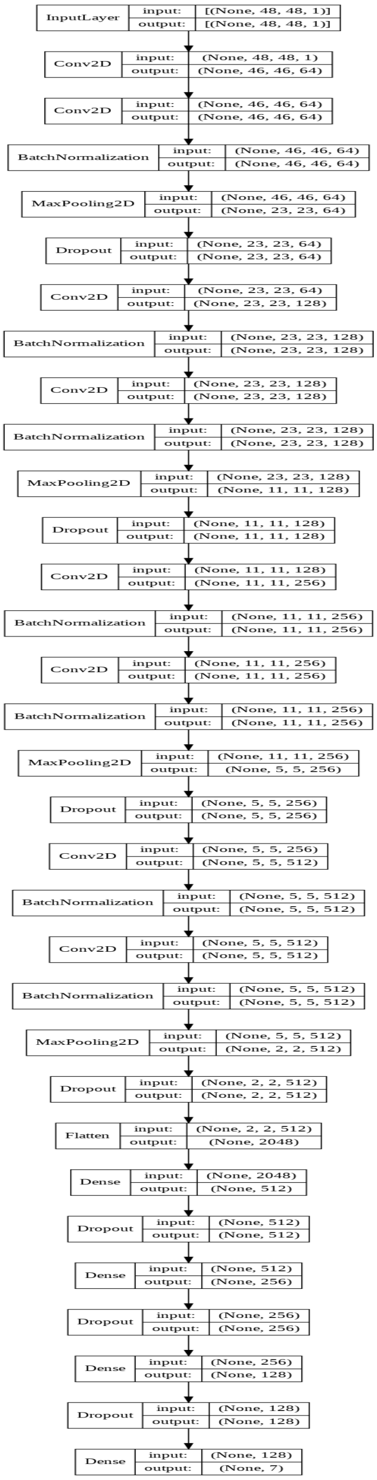
InputLayer input: [(None, 48, 48, 1)] output: [(None, 48, 48, 1)]
Conv2D input: (None, 48, 48, 1) output: (None, 46, 46, 64)
Conv2D input: (None, 46, 46, 64) output: (None, 46, 46, 64)
BatchNormalization input: (None, 46, 46, 64) output: (None, 46, 46, 64)
MaxPooling2D input: (None, 46, 46, 64) output: (None, 23, 23, 64)
Dropout input: (None, 23, 23, 64) output: (None, 23, 23, 64)
Conv2D input: (None, 23, 23, 64) output: (None, 23, 23, 128)
BatchNormalization input: (None, 23, 23, 128) output: (None, 23, 23, 128)
Conv2D input: (None, 23, 23, 128) output: (None, 23, 23, 128)
BatchNormalization input: (None, 23, 23, 128) output: (None, 23, 23, 128)
MaxPooling2D input: (None, 23, 23, 128) output: (None, 11, 11, 128)
Dropout input: (None, 11, 11, 128) output: (None, 11, 11, 128)
Conv2D input: (None, 11, 11, 128) output: (None, 11, 11, 256)
BatchNormalization input: (None, 11, 11, 256) output: (None, 11, 11, 256)
Conv2D input: (None, 11, 11, 256) output: (None, 11, 11, 256)
BatchNormalization input: (None, 11, 11, 256) output: (None, 11, 11, 256)
MaxPooling2D input: (None, 11, 11, 256) output: (None, 5, 5, 256)
Dropout input: (None, 5, 5, 256) output: (None, 5, 5, 256)
Conv2D input: (None, 5, 5, 256) output: (None, 5, 5, 512)
BatchNormalization input: (None, 5, 5, 512) output: (None, 5, 5, 512)
Conv2D input: (None, 5, 5, 512) output: (None, 5, 5, 512)
BatchNormalization input: (None, 5, 5, 512) output: (None, 5, 5, 512)
MaxPooling2D input: (None, 5, 5, 512) output: (None, 2, 2, 512)
Dropout input: (None, 2, 2, 512) output: (None, 2, 2, 512)
Flatten input: (None, 2, 2, 512) output: (None, 2048)
Dense input: (None, 2048) output: (None, 512)
Dropout input: (None, 512) output: (None, 512)
Dense input: (None, 512) output: (None, 256)
Dropout input: (None, 256) output: (None, 256)
Dense input: (None, 256) output: (None, 128)
Dropout input: (None, 128) output: (None, 128)
Dense input: (None, 128) output: (None, 7)

Figure 2. The Model Architecture

In order to achieve the best performance, I tried many models (from different sources) and settings, tuning up the hyper-parameters for each one of those. At first, I had a low accuracy of 60% and Val accuracy around the same, with a 12 layers model. So I increased epochs from 30 to 50, changed the batch size from 32 to 64, and I downsized the model, removing a couple of extra Conv2D layers from 12 to 10. I also changed the kernel size from 5x5 to 3x3, equalising the whole model. I got a better result, although still too far. So, eventually, I ended up with a 12 layer ConvNet (8 Hidden Layers and 4 Classification Layers). The first one starts with 64 filters, and then it multiplies that value by two as it goes in deeper. The rest of the hyper-parameters do not change along the way. Activation = Rectified Linear Unit (relu), kernel size: 3x3, MaxPooling2D with pool size: 2x2, with stride size: 2x2 (the size of the filter's steps/shifts) both applied from 2nd layer. To regularise the model, I used Ridge regression (L2) with value of 0.01, applied only at the first layer and then Dropout at 50% (40% over 10th and 11th layers). BatchNormalization, to standardise the inputs on the next layer, to stabilise the learning process, is also applied. I used Adam as optimiser with a learning rate = 0.001 and categorical cross-entropy as loss function. I then set epochs to 100, batch and size to 64. I tried to use the method Reduce Learning rate on Plateau, combining it with 125 epochs, but it did not improve much the learning process, as the dataset has its limits. As mentioned already, the FER2013 dataset is used to train the ConvNet, is read from a .csv file (which contains the raw pixels values for each image, associated with his label) for convenience in terms of time complexity, which is then split into three different sets: train, test and validation set. Once I noticed that the model was overfitting with long training sessions, I applied data Augmentation (on the train set). I managed to bring a little down validation loss (20%). As a result, I got a validation accuracy of roughly 66%. I trained this last model 4 times to check the different predictions generated by each training, loading the weights and comparing the result with the same image. The Val Accuracy from Tang, the winner of the FER2013 competition, is 71.2%[7], so I consider my result respectable.

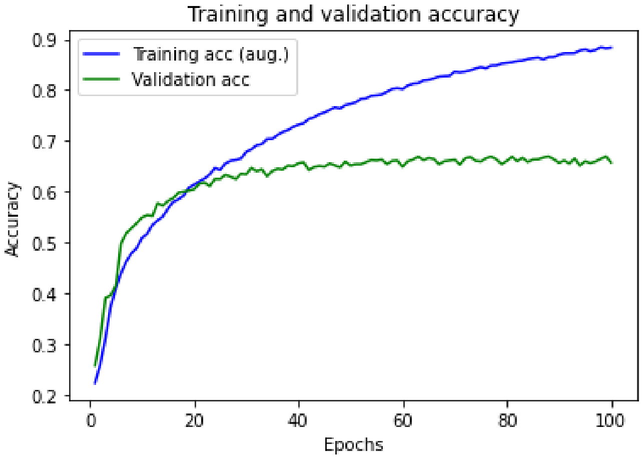

Figure 3. Training with data augmentation

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{1}$$

$$Precision = \frac{TP}{TP + FP} \tag{2}$$

4 Discussion

Comparing these results with other approaches such as[11][8][4][13], we can see better results with more complex (in some cases hybrid) models and better datasets to train with, such as CK+ or JAFFA. This paper [11] proves, in short, that using a more detailed dataset can make a big difference. They had the same accuracy as I have (approx. 66%) trained on a CNN, and using CK+ (which has a total of 5876 labelled images from 123 subjects), trained on a linear SVM though, they reached 98.4%.

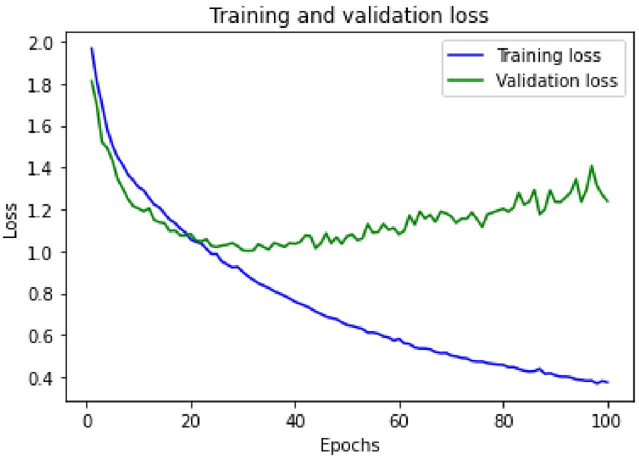

Figure 4. Training and Validation loss graphic

The most efficient that I found so far, in my opinion, comes from Stanford University, where these students have created a real-time simple layout web page for expression of emotion recognition [8]. In terms of practical efficiency is a very well made job; their results speak for themselves. They managed to reach 75.8% of accuracy on the FER2013 dataset, which is the highest score so far, overtaking Tang's 71.2% and the most recent Pramerdorfer's 75.2% [12] (all of those with CNNs). They then applied transfer learning with Keras VGG-Face library and pre-trained ResNet50, SeNet50 and VGG16 models. Great research.

This research from Imperial College London [9] outlines a detailed and complete approach to real-life expressions, where the datasets are "in-the-wild," i.e., from uncontrolled environments (e.g., group conversation in an open social space), defining a new error criterion for the training of DNN with these diverse datasets. They do this by using a combination of CNN and RNN architecture to improve generalization and versatility of network's knowledge, using Emotiw 2017 as dataset (comprehensive of video frames and images) and applying clustering a the last hidden layer, treating the expression classes as clusters. Then, using the cluster centroids to train the new CNN-RNN with a new dataset, they developed the appropriate loss function, reducing forgetting, so that transfer learning is a peak efficiency.

In addition, I then tested the trained ConvNet prediction with different images style, formats and features. I tested it with, out of context images, from non-neutral environments, including myself, also from different datasets such as CK+ and JAFFE. I tested the prediction using 3 different weights, obtained by 3 different trainings with the same hyper-parameters (100 epochs, etc.). As suspected, the easiest emotion to spot is Happiness, although, sometimes it gets confused with surprise, when the latter tend to express it more with the eyes regions (in particular with Asians). Similar confusion happens with sadness and neutral, sadness and fear, also between anger and fear. The most difficult expression to spot is disgust, having less than 500 images to train with, some of them not even very clear to the human vision. In addition, in the original dataset, in classes such as fear, disgust, angry, sad, neutral, we can spot some inaccuracy in those expressions, e.g., an angry image showing a neutral/sad expression.

5 Conclusion and future work

In the future I would like to try to train the model with a mixture of datasets, first checking the results of a single dataset with the same hyper-parameters, testing the prediction on "external" images. And then merging them into 1 big dataset to increase the originality of features across all the data, obviously standardising their size and

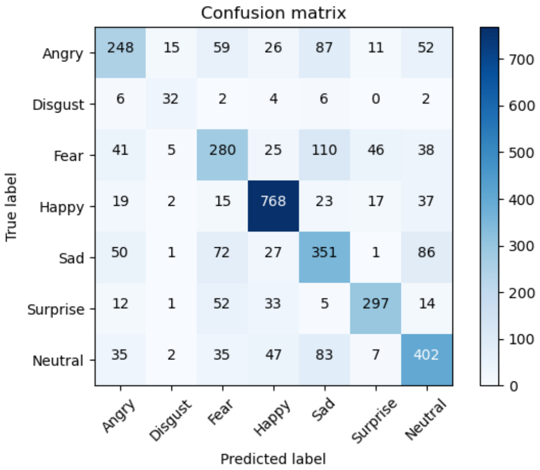

Figure 5. Confusion Matrix

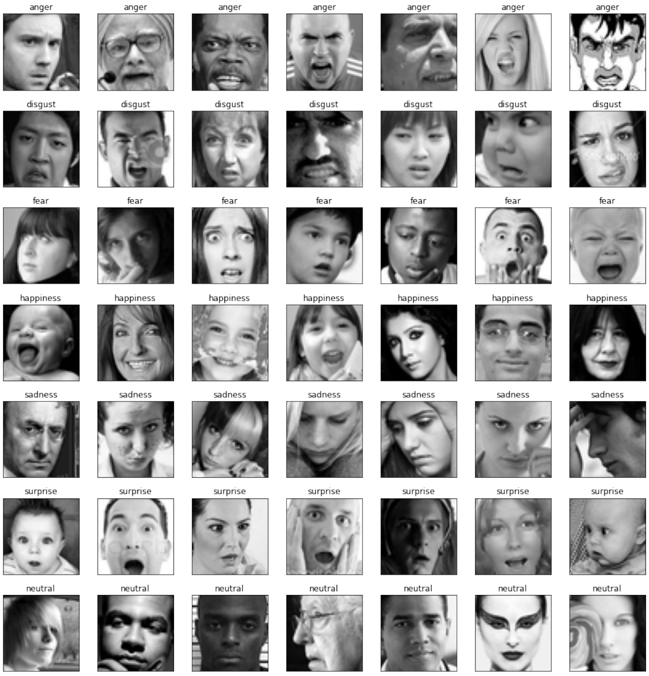

Figure 6. FER2013 dataset

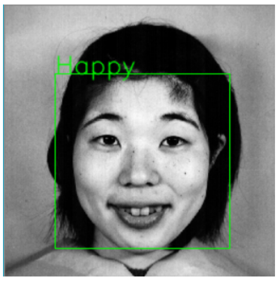

Figure 7. Happy prediction (True)

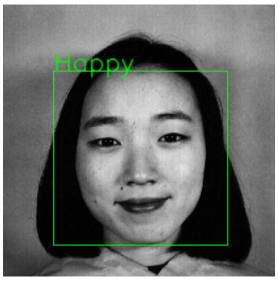

Figure 8. Happy prediction (True)

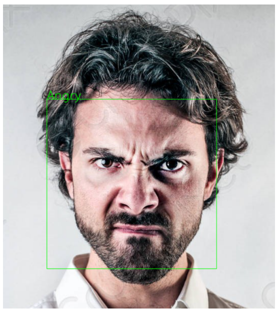

Figure 9. Angry prediction (True)

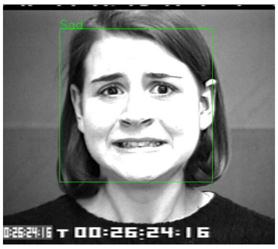

Figure 10. Pred: Sad, True: Fear

colours to make them suitable. I believe that the model will have a richer accuracy and versatility in predicting expressions from out-of-contex images. I also would like to include K-Fold cross-validation, which I found very interesting and capable to balance the learning process better.

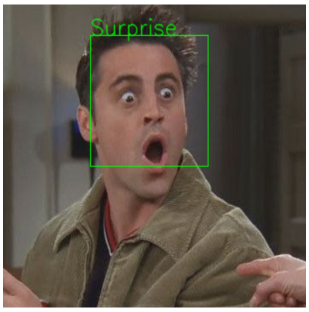

Figure 11. Surprise prediction (True)

Py IDE Enviroment Code PyCharm enviroment created on Anaconda, running Python 3.6 version, tensorflow-gpu 2.1.0, keras-gpu 2.3.1, CUDA 10.1.243, cudNN 7.6.5 versions.

REFERENCES

[1] Charles Darwin, *The expression of the emotions in man and animals*, New York ;D. Appleton and Co.,, 1916. https://www.biodiversitylibrary.org/bibliography/4820.

[2] Erin Digitale. Google glass helps kids with autism read facial expressions, 2018.

[3] P. Ekman, *Emotions Revealed: Recognizing Faces and Feelings to Improve Communication and Emotional Life*, Henry Holt and Company, 2004.

[4] Shima Alizadeh Azar Fazel, 'Convolutional neural networks for facial expression recognition', *b*, (2017).

[5] Douglas Heaven. Why faces don't always tell the truth about feelings, 2020.

[6] Rachael E Jack, Oliver GB Garrod, Hui Yu, Roberto Caldara, and Philippe G Schyns, 'Facial expressions of emotion are not culturally universal', *Proceedings of the National Academy of Sciences*, **109**(19), 7241–7244, (2012).

[7] Kaggle. Challenges in representation learning: Facial expression recognition challenge, 2013.

[8] Amil Khanzada, Charles Bai, and Ferhat Turker Celepcikay, 'Facial expression recognition with deep learning', *CoRR*, **abs/2004.11823**, (2020).

[9] Dimitrios Kollias and Stefanos Zafeiriou, 'Training deep neural networks with different datasets in-the-wild: The emotion recognition paradigm', *Contest*, 1–8, (2018).

[10] Runpeng Liu, Joseph P Salisbury, Arshya Vahabzadeh, and Ned T Sahin, 'Feasibility of an autism-focused augmented reality smartglasses

system for social communication and behavioral coaching', *Frontiers in pediatrics*, **5**, 145, (2017).

[11] Guilherme Reis Minh-An Quinn, Grant Sivesind, 'Real-time emotion recognition from facial expressions', *a*, (2017).

[12] Christopher Pramerdorfer and Martin Kampel, 'Facial expression recognition using convolutional neural networks: State of the art', *CoRR*, **abs/1612.02903**, (2016).

[13] Debnath Tanoy, Reza Md, Rahman Anichur, and Band Shahab, 'Four-layer convnet to facial emotion recognition with minimal epochs and the significance of data diversity', *psychology*, (05 2021).

[14] K. Vemou, T. Zerdick, A. Horvath, and European Data Protection Supervisor, *EDPS TechDispatch: Facial Emotion Recognition. Issue 1, 2021*, EDPS TechDispatch, Publications Office of the European Union, 2021.