Facial Expression Recognition with Quarantine Face Masks Using a Synthetic Dataset Generator

Yücel Çelik and Sezer Gören
Department of Computer Engineering, Yeditepe University, Istanbul, Turkey

Keywords: Facial Expression Detection, Face Masks, Covid-19, AI, Machine Learning, Convolutional Neural Network.

Abstract: The usage of face masks has increased dramatically in recent years due to the pandemic. This made many systems that depended on a full facial analysis not as accurate on faces that are covered with a face mask, which may lead to errors in the system. In this paper, we propose a Convolutional Neural Network (CNN) model that was trained solely on face masks to be more accurate and on point, that could more easily determine facial expressions. Our CNN model was trained with a seven different expression category dataset that only had people with face masks. Although we could not find a suitable dataset with face masks, we opted to generate a synthetic one. The dataset generation was done using Python and the help of the OpenCV library. The process is, after finding the dimensions of the face, we Perspective Transform the face mask object to be able to overlay it on the face. After that, the CNN model was also generated using Python with a CNN model. Using this method we gathered favorable results on the test subjects with 70.1% accuracy on the validation batch where previous facial expression recognition systems mostly failed to even recognize the face since they were not trained to recognize faces with face masks.

1 INTRODUCTION

FER (Facial Expression Recognition) **Figure 1** is the act of recognizing a person's mood from the various elements and mimics on their face such as the curvature of the mouth, position of their eyebrows, crow's feet at the edge of their eyes, and so forth. Being able to detect the changes stemming from many of these human mimics could open various opportunities for use in many fields. For example, in psychology it is critical that a patient's psychological status could be determined throughout any time of the day. To give another example, with FER it is possible to get automatic customer feedback for various services and to determine the disposition of the workers as well.

Figure 1: Facial expression examples (Singhal, 2021).

Admittedly, if the process of FER could be automated by a machine, this would enable incredible potential. Since, the machine could be able to pick up on many minute details a human would potentially miss, less prone to human errors, and take a fraction of the time to determine the result.

Fortunately, there have been many papers and work published on this topic and great advancement is made already. Remarkably, one of the first to publish a research paper goes back to 1992 (Kobayashi and Hara, 1992). In the aforementioned paper, the authors were trying to classify six distinct emotions with the help of a neural network, which is a type of machine that is modeled after the human brain to be able to think alike and made using many interconnected nodes and artificial neurons that process information, that eventually produce a result. Thereafter, much more work has come to light. Amongst those that have a focus on FER, most are trained on a person's face without obstruction and regrettably, those models that have been trained to work with an unmasked face might not be viable enough to be used in the modern world with so many face masks worn by people. In today's standards, since those models lose a remarkable amount of patterns and features when a person wears a face mask such as the mouth where

Çelik, Y. and Gören, S.
Facial Expression Recognition with Quarantine Face Masks Using a Synthetic Dataset Generator.
DOI: 10.5220/0011992500003497
In *Proceedings of the 3rd International Conference on Image Processing and Vision Engineering (IMPROVE 2023)*, pages 199-205
ISBN: 978-989-758-642-2; ISSN: 2795-4943

the model is specifically trained to take data from, they might not be working as intended. For instance, the AI model this paper (Sharifara et al., 2014) had developed, uses Haar Cascade to detect human faces but since it was developed without subjects with face masks in the dataset, it has trouble detecting subjects with face masks.

This is why the focus on this paper is how a CNN model could be used to detect facial expressions even with a face mask on the person's face. Our CNN model was specifically trained with a expression dataset that has been synthetically overlaid with face masks. The reason behind that is the lack of a usable and viable face masks FER dataset open to the public. There are few face mask dataset generators (Yang et al., 2020). In addition, with this dataset generator in mind, many models are going to have a much easier time finding a dataset in the future.

While generating our dataset to make a better attention map for our CNN model to focus, we have synthetically overlaid masks using OpenCV's homography methods to have good symmetry. The aforementioned methods are "getPerspectiveTransform" to make a transformation matrix and "warpPerspective" (DeTone et al., 2016) to apply the transformation matrix to the mask image. Randomly picked samples could be seen in **Figure 2**.

Figure 2: Facial expressions (Singhal, 2021) with synthetically added face masks.

After that, when all seven expressions containing the facial expressions are completed being overlaid with perspective transformed masks, the CNN model was trained over this dataset to generate a model. Since this process is automated by the Convolutional Neural Network (CNN) model, the system could potentially pick up many easy to miss subtleties from the subject's face from the view of a naked human eye.

Consequently, in this work after the training of our CNN model was done it could be plugged into any webcam to get real time detection of facial expressions with face masks. Since this process does not take too much resources it could be installed on most modern computer architectures to be used conveniently.

2 RELATED WORK

We have been able to find a similar work that has a FER system where it also focuses on facial expression detection with face masks (Yang et al., 2021). There are papers that only focus on real-time face detection (Yang et al., 2021), and papers that focus on detection of faces in real-time with face masks (Das et al., 2020).

If we were to compare this paper to similar work the closest one is by B. Yang, W. Jianming, et al. (Yang et al., 2021). In the aforementioned paper, the authors had used a binary classifier deep learning AI model to classify a FER and in this paper we worked with a CNN machine learning model. On the other hand, the paper one and our paper have dissimilarities. Our paper has seven expression categories that are 'Angry', 'Disgust', 'Fear', 'Happy', 'Neutral', 'Sad', 'Surprise' but the paper (Yang et al., 2021) has the three categories of 'Positive', 'Negative', 'Neutral'. Also, our paper was trained over a synthetically made dataset over a CNN model, while the mentioned paper (Yang et al., 2021) has been trained by deep learning binary classifier network.

One other similar work that has been published is (Castellano et al., 2022). It is a work that focuses on facial emotion recognition with face masks likewise but uses another dataset generation method and uses a deep learning model instead. Another similar paper on this topic is (Henke et al., 2022), which looks at the link between recognition of people with face masks of various ages. Henke, Lea, et al. achieve this goal by analyzing images with AI and drawing conclusions.

Lastly, one key difference between all referenced work is that our work's dataset has been entirely generated synthetically by our self-made masked image generator. More information regarding our dataset generator is in the following **section 3.1**.

3 PROPOSED METHOD

The steps we followed were starting with the conversion of the dataset, we fed the converted dataset into the model. After the CNN model's training is done, we send images in real-time from our webcam to be processed by our model to get a result. The dataset generator, CNN model, and real-time usage are run on a cloud environment using Google Colab.

3.1 Dataset Creation

The custom synthetic dataset has been crafted by overlaying the face mask image over the selected face.

Firstly, with the method of Facial Landmark Detection System (Wu et al., 2021) the coordinates of the facial elements. Example in **Figure 3** and their positions were recorded in a numpy (Van Der Walt et al., 2011) array to be used.

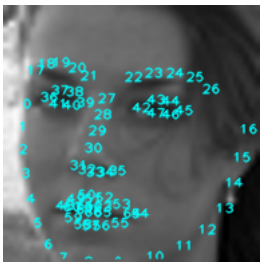

Figure 3: Facial landmarks.

With the coordinates taken, the ratio of these points was recorded so that the mask could be appropriately placed over the face. If no measures were taken and the mask is directly placed over the face some unwanted results were occurring. For example, if the face did not have any tilt vertically or horizontally the placement would not have any problems but if the face had a noticeable tilt at all the face masks that are overlaid would look out of place with the incorrect perspective as in **Figure 4a**. To fix this issue, the ratios of the face were calculated and some adjustments were made to the pivot points of the homography transformation according to the algorithm. Homography transformation is mostly used for straightening an image to each of its corners to 90° for most use cases. However, in this paper, the reverse of this process is applied to the image in question. The first input matrix is generated by giving the method to each of the corners of the original image. After that, four new pivot points are calculated using the aforementioned algorithm that takes the face's landmarks and calculates ratios using these points. After the calculated new points are given to the method a new transformation matrix is created which is applied on the face mask image to create a custom per-face face mask image ideal to the specific face that is being processed such as **Figure 4b**.

Afterwards, a set of other alterations are applied on the mask to make it possible to be overlaid. The first is cropping the image to the last calculated pivot points. This makes generation more efficient since this process lowers pixel amount, it takes less time per image transformation. This is also important since AI models require large amounts of reference images, we need to be able to convert them swiftly. The new cropped image only consists of the transformed mask, as seen in **Figure 5a**.

Following that, first a layer that consists of the face mask image's supposed alpha layer is created.

(a) Face with incorrect perspective mask applied.

(b) Face with the altered correct perspective mask applied.

Figure 4: Mask and face alignments.

This is done by creating a matrix sum of the 3 RGB (Red Green Blue) layers of the image using Numpy's "Sum" method. After that, each pixel in the alpha layer is multiplied by 255 since the original layer variables were doubles between 0 and 1, but the RGB values are 8-bit which is between 0 and 255. Afterwards, this layer is stacked on top of the original image with Numpy's 'Stack' method as the fourth alpha layer in the format RGBA (Red Green Blue Alpha), example in **Figure 5b**.

(a) Cropped mask.

(b) Alpha (transparency) layer added.

Figure 5: Image operations.

After those processes, we finally convert the image to the Pillow format from the OpenCV image format to be compatible with the newly opened face that is also on the Pillow image format and overlay the face mask image on top of the first pivot's x axis and y axis positions. The original face that has been used to find facial landmarks could have been used for this but we have decided to go with Pillow since it has better support for image overlay save features.

Ultimately, the face has been overlaid with a face mask that has been altered to fit over the face smoothly, such as **Figure 6**. We have tried to make it as realistic as possible, so that the AI model can pick features that are the most essential to get the best output without errors. Each generated emotion category has approximately between 2000 and 5000 images for the training of the CNN model and approximately between 500 and 1000 additional images for the trained CNN model to be tested for accuracy.

Figure 6: Different facial expressions (Singhal, 2021).

One of the reasons why we had gone this route was to have a dataset that had a good amount of images that is why we choose this facial expression dataset that has close to twenty thousand images and seven expression categories that are Angry, Disgust, Fear, Happy, Neutral, Sad, Surprise.

Therefore, to convert this entire dataset that had faces without face masks, we have converted the previously explained algorithm into a function that can be called repeatedly, defined pathways into both input and output folders and created a basic loop that can handle sub-categories to convert the dataset to people with face masks which marks the end of the dataset creation.

3.2 CNN Model Creation and Training

Required libraries are mentioned in the **section 3.4**. The dataset is fed to the program in the form of a zip file. After unzipping it we get seven different categories of emotions. After this for the training of the model, these seven categories of emotion are separated into their respective batches with the seven classes. After that the CNN of the model is created, it consists of 4 convolutional layers and after the flattening layer it has two dense layers and one final activation layer.

After the CNN model with the properties **Table 1** is created. We finally began training it. Where the dataset that is created passes by the model completely and where then the gradients of the images are backpropagated and the weights are updated.

Table 1: Layers used in the CNN model (Conv: Convolutional Layers, Flatten Layers, Connected Layers, Act: Activation Layers).

Layer Type	Conv.	Flatten	Connected	Act.
Num. of Layers	4	1	2	1

The dataset is first passed through the four convolutional layers. But first, the layers are initialized by the 'sequential' function. Only after that, the four convolution layers are passed through. Each convolution layer consists of five parts that are added to the model by the 'add' function. The first part is 'Conv2D' which specifies the layer that is going to be added to the model is a convolutional layer that also takes parameters for the size of the input and pooling. Pooling helps to cut the cost of processing cost of the image by precisely "pooling" the image into batches that are shrunken down. These batches are made of the maximum of the pixels or the average of the pixels. Secondly 'Batch Normalization' is added which applies a transformation that keeps the mean output close to zero and the standard deviation of the model close to one. Thirdly, the activation method is added to the layer. Which in this case is the "ReLU" (Linear Rectifier Unit) function. The Rectified Linear Unit activation function works by keeping the activation cost linear in the given value and always greater than zero. This helps with ever-increasing exponential growth of activation costs of neurons. The fourth part of the convolutional layer is the 'MaxPolling2D'. As the name suggests it works on 2D input and takes the best sample out of the group. Finally, 'Dropout' is added to the layer to prevent overfitting of the model. It works by negating the contributions of some neurons of the model and leaving others untouched. Through all four convolutional layers in the model, every part is kept the same except the 'Conv2D' method. The dimensions of the input and layers change throughout the four convolution layers in the model.

After the convolutional layers are done processing information, we flatten the input from the convolutional layers into a one-dimensional array for dense layers to be able to absorb it. The dense layers are where the classification of the images is made based on the previously calculated neurons and their weights. Following that, 'Batch Normalization layer', 'activation' and finally 'dropout layer' parts are added. Next to all this, we add a dense layer with its activation cost set to 'softmax' which is a method that converts the incoming numbers into probabilities so that we can get a reading of the classification. The classification is between the numbers '0' representing impossible and '1' representing total confidence in classification. The optimizer here used is the Adam Optimization Algorithm. It takes a parameter for learning rate. After that, the loss metric is added in the form of 'categorical_crossentropy' to determine the success of the model. After the CNN model is complete, with each epoch's pass the cur-

rently trained model with the weights and multipliers are saved. After that there is 'ReduceLROnPlateau' which adjusts the learning rate when a plateau or slow down in learning performance is detected. For example no accuracy increase for some number of epochs. The 'ModelCheckPoint' that allows to define checkpoints for model weights. Following that, 'Callbacks' with 'PlotLossesCallback' function to give reports in real-time about the training done. Finally, 'model.fit' method starts the training and validation of our model and we get **Figure 7** in each epoch.

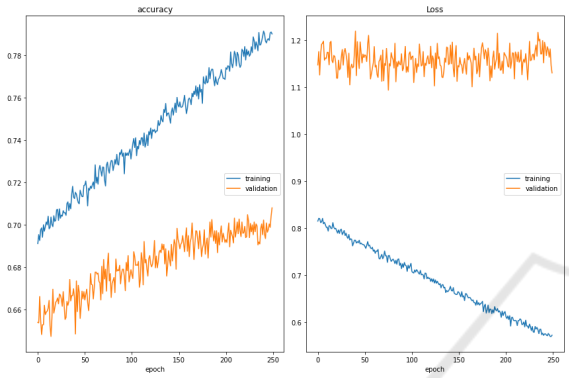

Figure 7: Accuracy and loss graph of test and validation batches up to 250 epochs.

From this figure, we can see a considerable improvement in the early epoch stages up to 150, where it starts to slow down and begin hitting a plateau on epoch 250. Also considering the trend of validation batch's loss line has started to take an upwards turn, we found it plausible to stop the training of the model to prevent any unintentional overfitting.

3.3 Real-Time Usage of the Trained Model

In this part, we are going to be focusing on how the newly trained model could be used to identify expressions on a person's face in real-time with a face mask.

With all the necessary libraries imported that are mentioned before we can continue. First of all, for the trained CNN model to be used the trained weights and multipliers are imported as 'FacialExpressionClass' to the system. In this class, the seven categories of emotions that we are working on are listed. After that, the imported weights are loaded into the model so that we can create a predictor model.

To interpret live-camera ,feed into it we take each image as JavaScript objects, decode and then encode them again as OpenCV image objects. It is important that the images are in the OpenCV image format so that we can run the model through them and overlay graphical elements over them such as the face box and most importantly, the detected expression.

We take the converted OpenCV format images and feed it through the Haar Cascade Classifier (Viola and Jones, 2004) first to filter the face part of the image or else the model would not work since it was trained to find expressions on a face with a mask not to find the face. After the Cascade filter takes out the face and we run the predictor through it we finally get the data that is going to be overlaid on the image. We do this by OpenCV's 'putText' function to insert the expression and 'rectangle' function to outline the face in the webcam live feed as can be seen in **Figure 8**.

So, the final product is a smooth continuous video-feed that shows what the current expression currently detected person is showing.

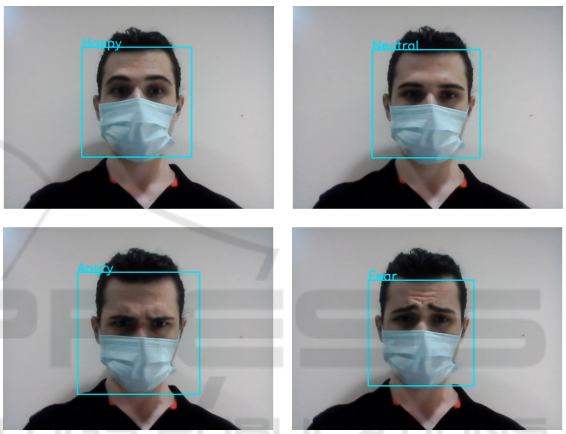

Figure 8: Real-Time usage.

3.4 Used Methodologies

For the IDE (Integrated Development Environment) and backend support, Google Colab has been used. The paper's coding has been coded entirely in Python version 3. For the unaltered dataset this (Singhal, 2021) has been used and later been transformed to the appropriate dataset using the algorithm that we had developed. For the mask perspective transformation algorithm the following libraries have been used: Numpy, OpenCV (Howse, 2013), google.colab.patches, dlib (King, 2009), Pillow (Clark et al., 2015), os Python modules.

As for the CNN usage and learning section, these libraries have been used: python-utils, livelossplot v0.5.2, Numpy, seaborn (Bisong, 2019), matplotlib.pyplot (Ari and Ustazhanov, 2014), os, tensorflow (with also keras API (Application Programming Interface) wrap (Brownlee, 2016)).

Lastly, for the real time FER detection these libraries have been used: IPython.display, google.colab.output, base64 (both encoding and

decoding), OpenCV (Howse, 2013), Numpy (Van Der Walt et al., 2011), Pillow (Clark et al., 2015), io, html, time (Christ et al., 2018)

4 TEST AND RESULTS

For the results, there have been some losses in accuracy due to parts of the facial features missing with the face masks. Not to mention, many mimics of the mouth are used to determine between many expressions but since it is covered some of the images have become quite difficult to tell even from a human eye's perspective. Despite that, the validation batch when training the model shows a value of 70.8% current accuracy at epoch 250. If we use the mode category for our baseline score, it would be 14.29%. On the other hand, our overall accuracy was 45.6%. When compared to the most similar paper (Yang et al., 2021). This paper has lower precision but when the categories are compared this paper has a 2.3 times larger classification range. To be exact, this paper features: 'Angry', 'Disgust', 'Fear, 'Happy', 'Neutral', 'Sad', 'Surprise'. The aforementioned paper (Yang et al., 2021) features: 'Positive', 'Neutral', 'Negative'. The statistics can be seen in **Table 2**.

Table 2: Accuracy and Loss for Both Training and Validation Sets.

	Training	**Validation**
Max Accuracy	79.1	70.8
Current Accuracy	79	70.8
Min Accuracy	69.1	64.7
Max Loss	0.821	1.131
Current Loss	0.572	1.220
Min Loss	0.596	1.094

As mentioned before in **section 2**, each expression category in the training set approximately has between 2000 and 5000 images and between 500 and 1000 in the training set. Making the training set to validation set ratio near 5:1.

Table 4 represents subjects with synthetically generated face masks made using our dataset generator and **Table 5** represents subjects that do not have a face mask. Both tables show results from subjects that are randomly picked and these values were found by running the CNN model and looking at the results in real-life not by simulation. We have included the two tables **Table 4** and **Table 5** to be able to have a grasp on our work does on both masked and unmasked faces.

The performance evaluation results are shown in **Table 3** and its data are based on **Table 4**. Formulas of the calculated values shown below. (TP: True Positive, FP: False Positive, TN: True Negative, FN: False Negative, P: Total Positive, N: Total Negative). The values calculated on **Table 3** are based on Eqn **1**, **2** and **3**.

$$Accuracy = (TP+TN)/(P+N) \quad (1)$$

$$Precision = (TP+TN)/(P+N) \quad (2)$$

$$Recall = TP/(TP+FN) \quad (3)$$

Table 3: Performance evaluation table (values shown are percentages. Ang: Anger, Dis: Disgust, Hap: Happy, Neu: Neutral, Surp: Surprise).

	Ang	Disg	Fear	Happ	Neut	Sad	Surp
accuracy	89.4	89.4	83.0	85.7	76.6	84.0	91.5
precision	50.0	100.0	40.0	58.3	45.5	33.3	25.0
recall	60.0	16.7	66.7	77.8	50.0	33.3	50.0

Table 4: Confusion matrix of subjects with face masks.

	Ang	Disg	Fear	Happ	Neut	Sad	Surp
Anger	50.0	0.0	16.7	11.1	0.0	0.0	0.0
Disgust	0.0	16.7	0.0	0.0	0.0	0.0	0.0
Fear	16.7	33.3	66.7	0.0	20.0	16.7	0.0
Happy	16.7	16.7	0.0	77.8	20.0	0.0	25.0
Neutral	0.0	0.0	0.0	11.1	50.0	50.0	50.0
Sad	16.7	33.3	0.0	0.0	10.0	33.3	0.0
Surprise	0.0	0.0	16.7	0.0	0.0	0.0	25.0

Table 5: Confusion matrix of subjects without face masks.

	Ang	Disg	Fear	Happ	Neut	Sad	Surp
Anger	50.0	0.0	16.7	11.1	0.0	0.0	0.0
Disgust	0.0	0.0	0.0	0.0	0.0	0.0	0.0
Fear	16.7	40.0	66.7	0.0	20.0	16.7	0.0
Happy	16.7	20.0	0.0	77.8	20.0	16.7	0.0
Neutral	0.0	0.0	0.0	11.1	50.0	50.0	50.0
Sad	16.7	40.0	0.0	0.0	10.0	33.3	0.0
Surprise	0.0	0.0	16.7	0.0	0.0	0.0	25.0

From these confusion matrices, it could be seen that some of the expression categories are much easier to classify compared to others. On a closer look, we could make out that expressions such as 'Disgust' are much harder to determine. We have ascertained that there are several factors as to why that is the case. One of which is, some expressions are especially distinguished from the mouth. For example, 'Disgust', 'Sad' or 'Surprise' emotions particularly lean on mouth mimics to convey oneself. Another reason is that, with the mouth region blocked, the boundary between the two emotions 'Anger' and 'Disgust' are nearly identical and proved to be incredibly hard to recognize from one another. Even from the standpoint of a human's perspective.

Also to improve the accuracy even higher, We have a few proposed ideas such as mask saturation randomization to eliminate bias for different illuminated environments or to expand the dataset. Not to mention, one of the state-of-the-art CNN models might yield completely different results and tests should be done to be determined.

5 CONCLUSION

Ultimately, in this paper, we had proposed a CNN artificial intelligence solution for automatic analysis of a person's facial expression and determination. Which had seven different expression categories that are 'Angry', 'Disgust', 'Fear', 'Happy', 'Neutral', 'Sad' and lastly 'Surprise'.

One reason so many expression categories have been used is to have a paper that is not too similar to others. Even though some expressions are incredibly hard to make out with a face mask even from a human's perspective. Although, reducing the number of emotions would have a dramatic increase in test accuracy, we would not want to downgrade our goal. To accomplish the objectives in this paper we have used Python in tandem with Google Colab and these two combined to offer an incredible workspace for developers.

The paper consisted of three main parts, the synthetic dataset generation since there is a lack of masked face datasets as of writing this. The CNN model was built and trained with our synthetically made dataset that we generated by using a pre-existing expression dataset with no masks. Subsequently, generating and saving the trained model's weights and multipliers. Ultimately, to be used in a video feed to offer real-time FER with face masks.

6 FUTURE WORK

We do believe that this work still has more space to grow. More features could be added to improve already existing systems. One other system that would be beneficial to add for example would be saturation randomization for the face masks in the dataset generator. Since the dataset consists of only black or white masks, the model might have a slight bias towards those two colors. However with a saturation randomization system it would be equivalent to color randomization since the images are processed in a single gray-scale layer, a saturation change would be comparable to a hue change in a full-color image.

REFERENCES

Ari, N. and Ustazhanov, M. (2014). Matplotlib in python. In *2014 11th International Conference on Electronics, Computer and Computation (ICECCO)*. IEEE.

Bisong, E. (2019). *Building machine learning and deep learning models on Google cloud platform*. Springer.

Brownlee, J. (2016). *Deep learning with Python: develop deep learning models on Theano and TensorFlow using Keras*. Machine Learning Mastery.

Castellano, G., De Carolis, B., and Macchiarulo, N. (2022). Automatic facial emotion recognition at the covid-19 pandemic time.

Christ, M., Braun, N., Neuffer, J., and Kempa-Liehr, A. W. (2018). Time series feature extraction on basis of scalable hypothesis tests (tsfresh–a python package). *Neurocomputing*.

Clark, A. et al. (2015). Pillow (pil fork) documentation. *readthedocs*.

Das, A., Wasif Ansari, M., and Basak, R. (2020). Covid-19 face mask detection using tensorflow, keras and opencv. In *2020 IEEE 17th India Council International Conference (INDICON)*.

DeTone, D., Malisiewicz, T., and Rabinovich, A. (2016). Deep image homography estimation. *CoRR*, abs/1606.03798.

Henke, L., Guseva, M., Wagemans, K., Pischedda, D., Haynes, J.-D., Jahn, G., and Anders, S. (2022). Surgical face masks do not impair the decoding of facial expressions of negative affect more severely in older than in younger adults.

Howse, J. (2013). *OpenCV computer vision with python*. Packt Publishing Birmingham.

King, D. E. (2009). Dlib-ml: A machine learning toolkit. *The Journal of Machine Learning Research*.

Kobayashi, H. and Hara, F. (1992). Recognition of six basic facial expression and their strength by neural network. In *[1992] Proceedings IEEE International Workshop on Robot and Human Communication*. IEEE.

Sharifara, A., Mohd Rahim, M. S., and Anisi, Y. (2014). A general review of human face detection including a study of neural networks and haar feature-based cascade classifier in face detection. In *2014 International Symposium on Biometrics and Security Technologies (ISBAST)*.

Singhal, A. (2021). *Facial Expression Dataset*. Kaggle.

Van Der Walt, S., Colbert, S. C., and Varoquaux, G. (2011). The numpy array: a structure for efficient numerical computation. *Computing in science & engineering*.

Viola, P. and Jones, M. J. (2004). Robust real-time face detection. *International Journal of Computer Vision*.

Wu, B.-F., Chen, B.-R., and Hsu, C.-F. (2021). Design of a facial landmark detection system using a dynamic optical flow approach. *IEEE Access*.

Yang, B., Jianming, W., and Hattori, G. (2021). Face mask aware robust facial expression recognition during the covid-19 pandemic. In *2021 IEEE International Conference on Image Processing (ICIP)*.

Yang, B., Wu, J., and Hattori, G. (2020). Facial expression recognition with the advent of face masks. MUM '20.