Facial Landmark Detection

Deep Learning

Alaa Noor
University of Potsdam
alaa.noor@uni-potsdam.de

Aziha Kamal
University of Potsdam
aziha.kamal@uni-potsdam.de

Murphy Sünnenwold
University of Potsdam
suennenwold@uni-potsdam.de

Contents

Abstract

The performance of facial landmark detection algorithms on static photos has recently been astounding. In videos with motion blur, these algorithms, however, are not very reliable and lack accuracy. Moreover, modern facial landmark detection algorithms struggle to produce factual findings because of the absence of structural information. In this work, we implement a Convolutional Neural Network (CNN) model for recognizing facial landmarks based on images of faces and apply it to video inputs using the Viola-Jones algorithm for face detection. The performance of the model is evaluated on the training dataset from the Kaggle Facial Keypoints Detection competition itself, which consists of 7049 labeled images and on labeled videos from the 300-VW dataset. Our experimental results show an accuracy of about 91% on the dataset and a 65% accuracy with an absolute error of 2.9 for selected videos from the 300-VW dataset that match our intended application of detecting the facial landmarks on webcam input.

1 Introduction

In visual communication, the face plays a vital role. Humans can interpret a great deal about another person's intent, identity, and emotion just by looking at their face. In computer vision, the localization of the fiducial facial keypoints is typically a crucial stage in the automatic extraction of that facial information. Many facial analysis techniques such as face alignment, face modeling, face recognition, head pose tracking, facial expression tracking or recognition, face verification or authentication, gender or age recognition etc [31], are constructed depending heavily on the information provided by these facial keypoints. For facial recognition, to "frontalize" the face and help eliminate the large within-subject differences and increase recognition accuracy, landmark positions on a 2D image are typically integrated with a 3D head model.

Facial landmark detection algorithms strive to recognize the locations of the facial landmark points automatically on videos or images. The keypoints are mainly the dominant points that describe either the specific location of a face component, such as the corner of the eye, or an interpolated point that connects the dominant points surrounding the facial contour and facial component. However, facial landmark detection still remains a challenging problem for several reasons although many efforts have been made lately. Even the state-of-the-art algorithms fail to identify the landmarks properly, due to presence of facial occlusions, different expressions or extreme pose variations, and different illumination settings.

Predicting facial landmarks on live webcam input is the goal of this work. This is accomplished by first training a CNN model on a collection of face photos including the x- and y-coordinates of 15 important facial landmarks to predict these coordinates in a given image, and then extending to conduct facial landmark detection on a live video using face detection and the model. Following the major facial landmarks as the subject moves in real-time is required once the face has been recognized.

2 Related Work

For the purpose of detecting facial landmarks, several traditional methods have been put forth in the literature during the past few decades. As component detectors, numerous individuals implemented SVM [2], Adaboost [22], or random forest [1] classifiers, and detection was based on local picture features. Methods for detecting facial landmarks are classified into two types: regression-based methods and template fitting methods.

Regression-based methods often use picture attributes to explicitly estimate landmark locations through regression. With random forests and support vector regressors, respectively, Dantone et al. [10] and Valstar et al. [30] predicted facial landmark points from local patches using regression-based methods. Cao et al. [5] and Burgos-Artizzu et al. [3] both used pixel-difference features in cascaded fern regression. Random regression forest was used by Cootes et al. [7] and Yang et al. [34] to determine the location of a landmark using a local picture patch containing Haar-like features.

On the other hand, during training, template fitting methods try to learn a shape model and fit input photos during testing. The original template fitting methods were developed by ASM [9] and AAM [8]. The shape of the face is represented in ASM [9] by a linear combination of fundamental shapes learned by PCA, and the appearance of the face is modeled by several pre-trained templates. When it comes to AAM [8], the shape representation is comparable to that of ASM, while the appearance is described by PCA using a regular coordinate system to avoid shape alterations. Based on the template fitting methods, face detection, facial landmark detection, and pose estimation can all be handled simultaneously, as demonstrated by Zhu and Ramanan [37].

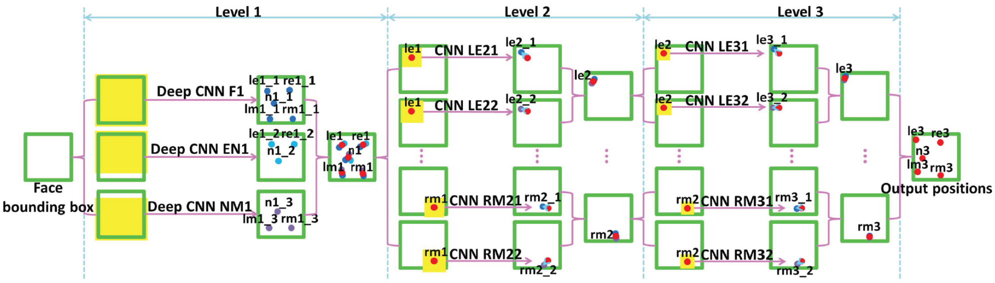

Figure 1: Three-level cascaded convolutional networks proposed by Sun et al. [26]

Deep learning algorithms significantly improve the accuracy of face landmark detection as they are used more and more in computer vision tasks. These techniques frequently frame the landmark task as a regression issue and the strategy used most frequently is coarse-to-fine. Typically, cascaded CNN models are used to achieve this, or an initial prediction and image pixel input is used. Sun et al. [26] have presented the algorithm of cascaded models that is most representative. A pre-partitioning of face pictures into various parts is necessary for the cascaded CNN, and each of these portions is processed by a distinct deep CNN. The outputs are then averaged and routed to several cascaded layers in order to process each facial landmark separately (Figure 1).

A multi-task learning approach for both face attribute estimation for example gender, smiling, wearing glasses, and other attributes) and five-points landmark identification was proposed by Zhang et al. [35, 36] in addition to coarse-to-fine methods.The position conditioned dendritic convolution neural network (PCD-CNN) [20], developed most recently by Kumar and Chellapa, combines a classification network with a second, modular classification network to increase detection accuracy. In their groundbreaking work, Duffner and Garcia [13] established what is now regarded as a straightforward neural network architecture, with just six layers total input and output. Comparing its application to the other earlier techniques, it performed remarkably well in situations with changing brightness and stance.

3 Methodology

3.1 Dataset

For our work, we are using a dataset from the Kaggle Facial Keypoints Detection competition [12]. The dataset consists of 7049 training images with labels and 1783 additional images without labels. Each of size 96x96 pixels in grayscale (0-255). Each keypoint is given as a (x,y) real-valued pair between 0 and 96. The following components of the face are represented by the 15 keypoints:

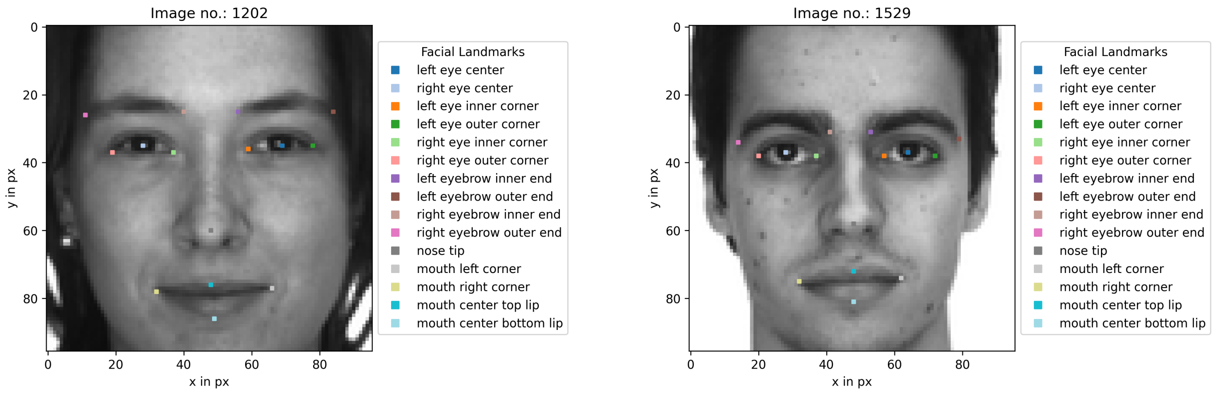

Figure 2: Two examples of fully labeled images.

In this context, left and right allude to the subject's point of view. In total about 50% of all the labels are missing. The dataset seems to originate from two different sources and can therefore be divided into 2140 clean datapoints which are fully labeled (Figure 2) and 4909 unclean datapoints which mostly only have 4 of the 15 landmarks (Figure 3). For further computations and training all the missing labels are filled with -1. The preprocessed data is also stored in fast accessible numpy files.

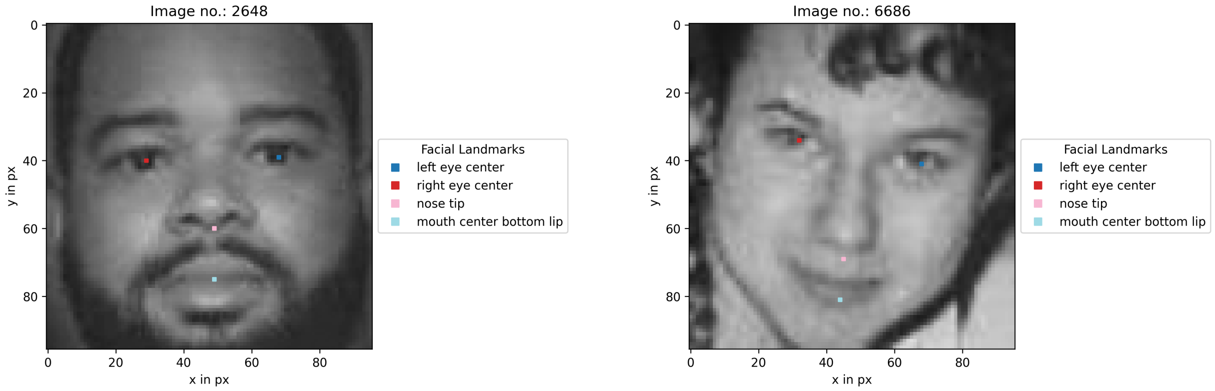

Figure 3: Two examples of partly labeled images.

Not only are a big part of the labels missing but the available labels often lack a lot of accuracy (Figure 4). Especially the ones from the unclean data. Most inaccurate labeled by far are the nose tip and the center of the bottom lip.

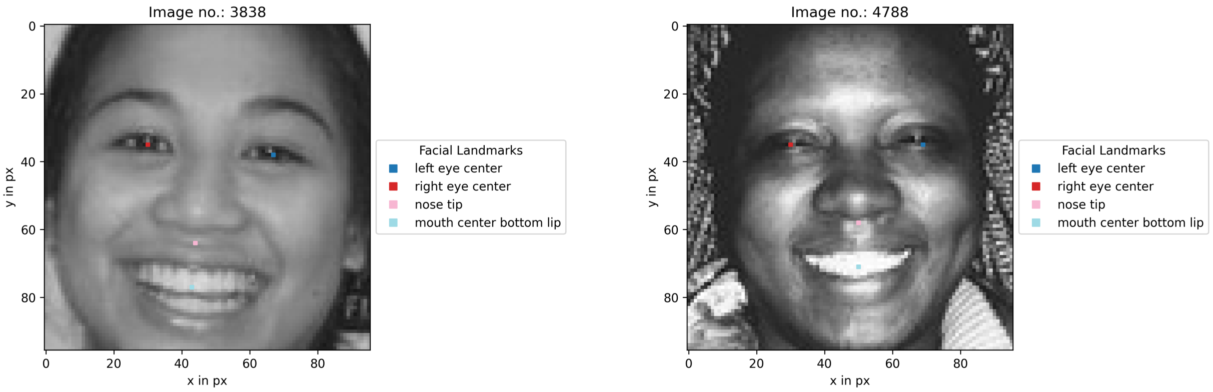

Figure 4: Two examples of poorly labeled images.

Most of the images are shot from a frontal and horizontal perspective without any tilt and the majority the of people look directly into the camera. Sadly the dataset is lacking diversity with only a few representatives of BIPoC.

3.1.1 Data augmentations

Generating augmented images allows to have more and different data to train on without the need for collecting new labeled data. Data augmentations help with overfitting and enables to alter the data to better align with the intended application of the model. Photometric distortion and geometric distortion are two distinct data augmentation techniques. Photometric distortion involves changing the contrast, brightness, saturation, hue, and noise of an image. Whereas geometric distortion is mainly about random scaling, flipping, rotation, cropping, and translation etc. To get more out of the dataset we have used a handful of data augmentations. The main goal is to reliably make good predictions on webcam inputs. Therefore we applied rotation, brightness and contrast, cropping and padding, horizontal flipping and minor perspective augmentations to the whole clean data individually to imitate real situations with the data available. This adds $5 \cdot 2140 = 10700$ datapoints to the dataset. After that in total the dataset consists of 17749 images. Most of the augmentations are rather minor. They are supposed to be realistic. For the augmentations of the images and their labels we used the python library Albumentations [4].

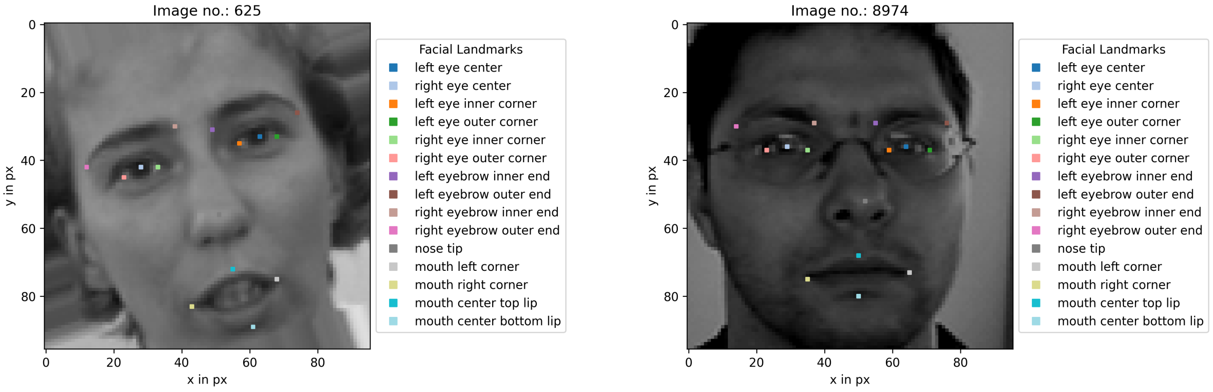

Figure 5: Two examples of augmented images (left: rotated, right: contrast and brightness).

3.2 Model

Convolutional Neural Networks are widely used for tasks ranging from image and video recognition, classification and analysis to recommendation systems, time series analysis, natural language processing, and speech recognition. CNNs offer two key advantages over traditional neural networks: a reduced number of weights makes them more effective in terms of memory and complexity while also allowing them to learn significant features from the input data at various levels [19]. For the purpose of our work, we use a CNN model. It takes 9216 float32 values in the shape of $96 \times 96 \times 1$ as input and outputs 30 float32 values.

3.2.1 Architecture

For predicting the facial landmark points from the images, we use a rather simple CNN model with few modifications in the layers. Our CNN model has 11 convolution layers each with same padding and without a bias. Each convolution is followed by a Rectified Linear Unit (ReLU) (alpha: 0.1) and a Batch Normalization. After every two convolution layers there is one max-pooling layer. And finally to get the 30 output nodes we flatten everything and use a dense neural network with a dropout with rate 0.1. The model architecture was inspired by different other architectures for this problem from the kaggle competition but mainly Karan Jakhar's approach [11].

The major role of the convolution is feature extraction. It is a mathematical operation to combine two sets of information. A feature map is produced by applying the convolution to the input data using a convolution filter. We reduced the number of parameters in the network by using 3x3 convolution. Next as an activation function we use the ReLU. Since traditional activation functions such as Sigmoid and Tanh produce vanishing gradient, whereas the unsaturated nonlinear attributes of ReLU are efficient to alleviate the problem [33]. After each ReLU, we use a Batch Normalization layer (proposed by Ioffe and Szegedy, 2015 [18]). Through a normalization step that fixes the means and variances of layer inputs, the Batch Normalization layer aids in reducing the internal covariate shift of the network. Here, the computations of mean and variance are made after each mini-batch rather than throughout the entire training set. Batch normalization also facilitates the use of increased learning rates without any risk of divergence, by reducing the dependence of gradient on it's initial values or on the size of the parameters [18]. Pooling is done after the convolution operation. Pooling layer down-sample the feature maps independently, by maintaining the depth while lowering the width and height. Pooling helps in reducing the dimensionality, this enables in shortening the training time and reduce overfitting. We used max-pooling in our model, which basically takes the maximum value in the pooling window.

Finally the output from the last pooling layer is the output to the Fully Connected layer. A Fully Connected layer is a feed-forward neural network. Before passing the input to the Fully Connected layer it needs to be flattened from a multidimensional matrix to a one-dimensional vector [6] [24] using Flatten layer and then fed to the Fully Connected layer. Lastly as a regularization technique, Dropout (proposed by Srivastava et al. 2014 [25]) layer is used for preventing overfitting in the network, as Fully Connected layer increase the number of parameters and they are more prone to overfit themselves by excessively co-adapting. Finally as the whole architecture contains several hidden layers it is also known as Deep Neural Network.

3.2.2 Optimizer, Loss, Metrics

For training the CNN the in keras built-in pre-implemented Adam optimizer is used which is a stochastic gradient descent method that is based on adaptive estimation of first-order and second-order moments. Since we are dealing with specific distances between the predicted coordinates and the true coordinates we are using the Mean Squared Error (MSE) as the loss function for the training. Additionally to asses the model during training the mean absolute error and accuracy metrics are also used. The accuracy thereby classifies a prediction to be correct when its position is within a radius of 2 from the true coordinates.

$$\|\hat{y} - y\| < 2$$

3.2.3 Masking

Since nearly 50% of the labels are missing, dealing with the unclean data is very important. The model should still train on the unclean data to learn the few landmarks that are labeled. To only train on the landmarks available we are masking the output during the calculation of the loss and metrics. The model should not learn to predict our filling value of -1 when the new input resembles the unclean data. Instead the model should always predict the location of all 15 landmarks. For that we ignore all the predictions where the true label is -1 (meaning that there is no label for that landmark) for calculating the loss and metrics.

Mathematically written ($\hat{y} \in \mathbb{R}^n$ true labels, $y \in \mathbb{R}^n$ predictions):

Masked Mean Squared Error

$$l = \frac{\sum^{n}_{\substack{i=0 \\ \hat{y}_i \neq -1}} |\hat{y} - y|_i^2}{\sum^{n}_{\substack{i=0 \\ \hat{y}_i \neq -1}} 1}$$

Masked Mean Absolute Error

$$l = \frac{\sum^{n}_{\substack{i=0 \\ \hat{y}_i \neq -1}} |\hat{y} - y|_i}{\sum^{n}_{\substack{i=0 \\ \hat{y}_i \neq -1}} 1}$$

3.2.4 Training

Our work resulted in four major versions of the model. The latest one version 4 is the best performing one. It was trained in 40 minutes on a Nvidia Tesla T4 on the Google Cloud over Google Colab. V4 was trained on the whole dataset including the unclean data and all the augmented data with the following hyperparameters chosen from experience from the last versions:

Hyperparameter	Value
Epochs	100
Validation split	20%
Batch size	256
ADAM learning rate	0.001
Number of images	17749

Besides has all the recent research on CNNs shown that they highly benefit from more training data. The training is kept rather short and the training history shows no sign of over- or underfitting.

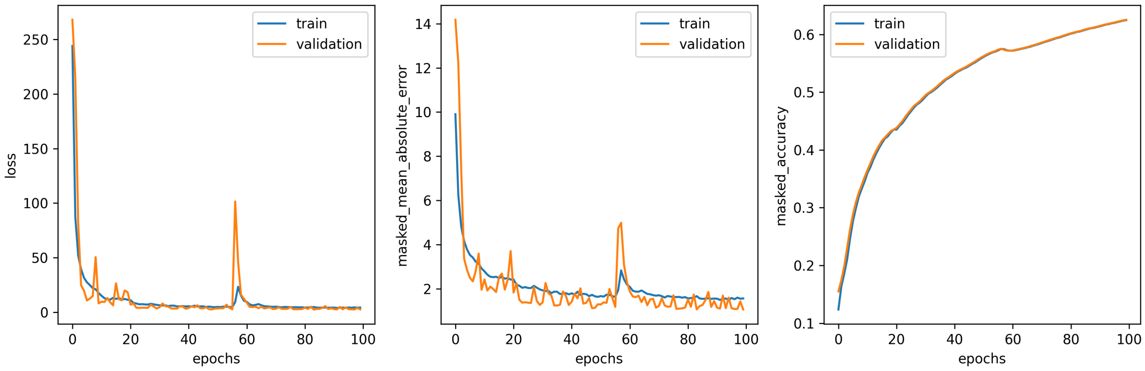

Figure 6: Training history of the fourth version of model.

The training finished with the following results:

masked mean squared error	4.25
masked mean absolute error	1.56
masked accuracy ($\|\hat{y} - y\| < 2$)	62.5%

Over all the different versions we experimented with different amounts of the unclean data to train on and different amounts of data augmentations. Because we are working with a CNN and are masking the output more data seems to be generally better. Training with more than 100 epochs with the unclean data effects the performance negatively. The model overfits and the unclean data becomes a problem. All the missing landmarks do not effect the loss and for images that resemble the unclean data the model learns to just clump all the points together in the middle of the face.

Table 1 shows how the different versions of our model are performing based on various epochs and number of images. The batch size and the validation split for all versions are 256 and 0.2, respectively. Starting with the model version 2 two key advantages over version 1 are that the training goes way faster and during the training the loss is masking the output for missing values. On the other hand, rotation-augmented data were used to train V3, with a slightly bigger dataset. Finally, more augmentation like rotation, horizontal flip, cropping, padding, perspective, brightness, and contrast was added to V4's training. Except for version V1, all the versions were trained on the Colab GPU. The epoch size for V1 was also considerably lower than that of the other versions, and it was trained using a Ryzen 5 3600 processor, which increased the processing time immensely.

Model	Epochs	Total Images	Time	mmse	mmae	Accuracy
V1	20	7049	43 min	-	-	-
V2	100	7049	15 min	4.3127	1.584	0.5411
V3	100	11329	25 min	3.7041	1.4212	0.6283
V4	100	17749	40 min	4.2459	1.5622	0.6248

Table 1: Comparison of different model versions.

3.3 Applying the Model to Video Inputs

The model was only trained on images of faces. The face always filled out the whole image. To apply the model to video input it is first necessary to find the location and size of a face in a given frame and bringing it into shape to predict on. The model is meant to only predict on the faces and not arbitrary images with different backgrounds.

3.3.1 Viola-Jones Algorithm

For detecting the face in real time we use Viola-Jones' algorithm [32]. Viola-Jones' is a face detection algorithm that was proposed jointly by Paul Viola and Michael Jones. It is based on three main concepts Integral Image (uses Haar Feature Selection named after Alfred Haar), AdaBoost and Cascading classifiers.
First a learning algorithm needs to be applied to identify the face. The image frame is divided into a segment of rectangles. These rectangles are then used by Haar Feature Selection to detect features using detection windows in the image fragment. The integral image is then constructed by adding the values in a rectangular segment of the grid. The equation for integral image is given by:

$$ii = \sum_{\substack{x' \leq x \\ y' \leq y}} = i(x', y'),$$

As the features of rectangle is adjoining to at least another rectangle, sum of any rectangle feature can be calculated in four array references using the integral image which can be seen in (Figure 7.a). It would be 6 for two rectangles features, 8 for three rectangles features and so on as shown as in (Figure 7.b). The main advantage of the integral image is that it makes the calculations much less intensive and save computation time for any facial detection model.

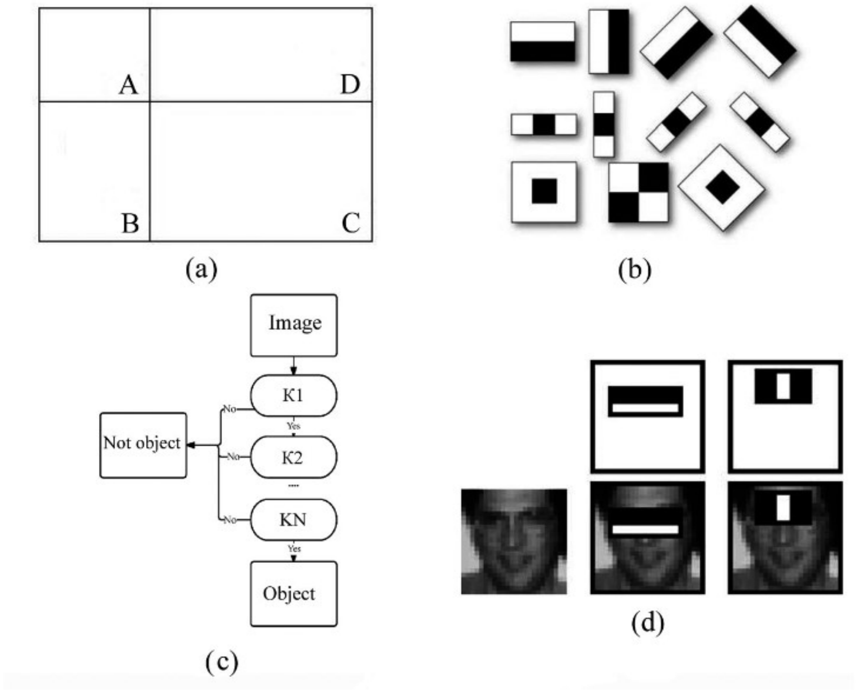

Figure 7: Viola-Jones algorithm parts: (a) combination of regions, (b) Haar Features, (c) cascade classifier, and (d) Haarfeature application to the image. [14]

Moving on to training, small number of important features is then selected using AdaBoost [15] to construct the classifier. Viola Jones slightly modified the original AdaBoost algorithm [27]. It combines classifier series as a sequence of filters as displayed in (Figure 7.c) which efficiently classifies image regions. Each Haar-like feature act as a weak learner. Before a feature is included to the final classifier, AdaBoost evaluates the performance of each classifier. The performance is assessed on all sub-regions of image used for training. Certain sub-regions deliver a strong reaction to the classifier. Those are recognized as positives, which the classifier thinks it as a human face. Whereas sub-regions which do not deliver strong reaction are identified as negatives, as it does not contain human face according to the classifier. Classifiers with the best performance are given more weight or relevance. The outcome is a strong classifier, also known as a boosted classifier, which incorporates the best weak classifiers. Basically the Adaboost algorithm used here sets a minimum threshold to identify whether a classified feature is useful or not [28].

Lastly, Cascading classifier is another sort of "hack" to boost the speed and increases performance of the model. It allows background regions of the image to be quickly discarded while spending more computation on promising face-like regions. For example, an image is discarded immediately for not having a face-like region if it receives a negative evaluation in the first stage. The cascading classifier can discard an image at any stage which thereby increases the necessary speed for real-time face detection [23] [28].

4 Evaluation

4.1 On the Dataset

On the dataset itself the model performs very well. It is similar to what other developers achieve with the same dataset. There is no significant difference between the performance on the dataset with the unclean data included and not.

Without unclean data (12840 images):

masked mean squared error	2.3
masked mean absolute error	1
masked accuracy ($\|\hat{y} - y\| < 2$)	90.2%

With unclean data (17749 images):

masked mean squared error	2
masked mean absolute error	1
masked accuracy ($\|\hat{y} - y\| < 2$)	91.8%

For most images the performance looks like this:

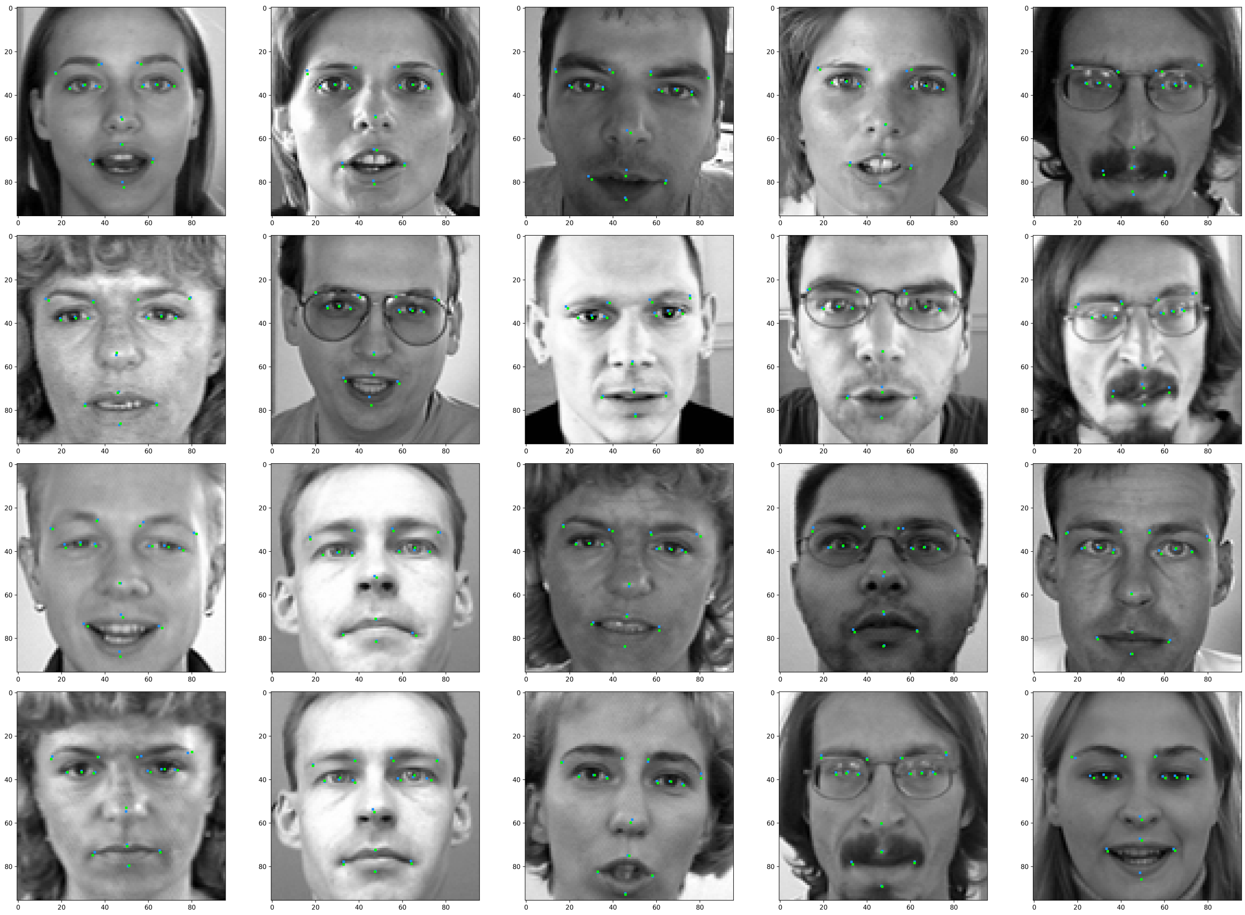

Figure 8: Model prediction on 20 images from the dataset.

The accuracy is very high and the predicted landmarks fit to the image. Although for some other images that is of course not the case. Many times the nose tip will be predicted below its actual location and the model cannot really deal with open mouths. Not because of the bad performance of the model but the inaccurate labels of many of the images.

4.2 On Video Inputs

But the performance on the dataset is rather secondary to us though. The goal is to use the model on live webcam input. To evaluate on actual labeled videos we use labeled videos from the 300-VW dataset [16][17][29]. The videos are all roughly one minute long at about 25-30 frames per second. The dataset has labels for 68 landmarks. Not all of the 15 landmarks from the original training dataset are included in these 68 but are deductible from surrounding keypoints. For the evaluation we picked four videos that resemble the situation we were working towards the most that shows the people in front of a webcam or with a similar perspective and distance. The selected videos are the videos with numbers: 39, 223, 224 and 406. From every video are 20% of the frames randomly selected. Then we used the Viola-Jones algorithm on those to detect the faces and predict on them. The results can then be mapped back to the full size of the frame. For the 3174 frames from the four videos we get the following results:

masked mean squared error	34.7
masked mean absolute error	2.9
masked accuracy ($\|\hat{y} - y\| < 2$)	65%

The high squared error implies that there are probably some bigger outliers but the overall performance with an accuracy of 65% and an absolute error of 2.9 is good. This can also be shown on the following two examples:

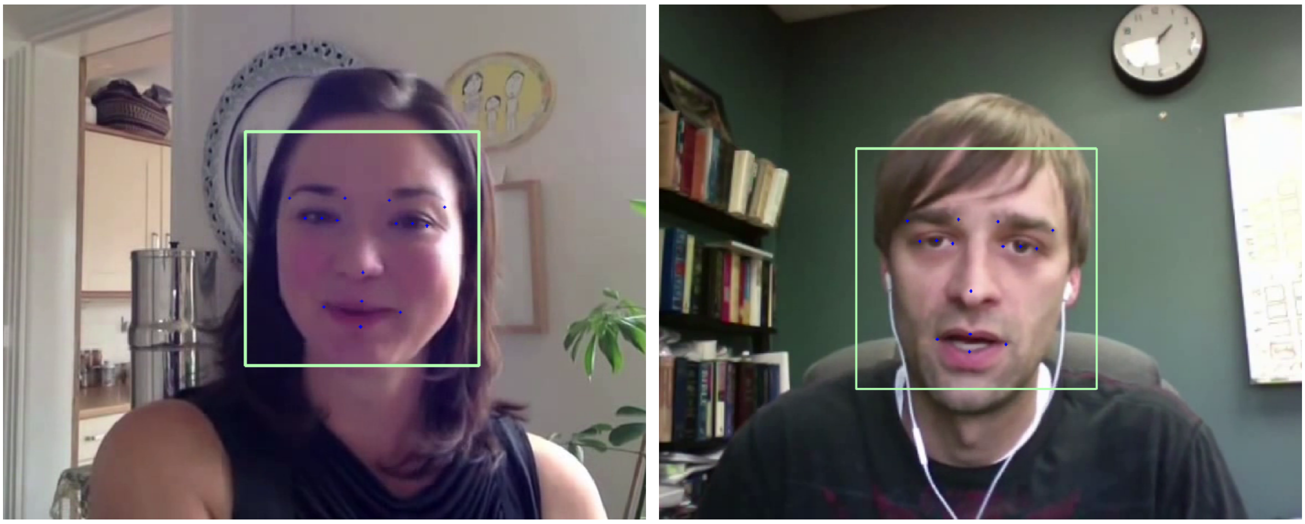

Figure 9: Two examples of model predictions on videos.

At first 65% may not seem like a lot but given the condition of a value being predicted correctly being rather strict being able to very accurately predict 2/3 of the given landmarks is really good.

5 Discussion

Overall we the model performs well and the project was successful. With what we have right now in terms of the dataset and model architecture there is nearly no room left to grow. A lot of time went into optimizing the execution speeds. When dealing with a lot of data the pre-processing, loading, storing and the whole infrastructure plays a crucial role. That part went very well and we build a solid foundation to apply our computer vision model to videos.

5.1 Limitations

To even further increase the performance of the model to detect facial landmarks in video inputs there are some major limitations to overcome. Most of which were already mentioned. By far the biggest drawback in our experiment was the dataset itself. In total, 50% of all the labels are missing, which makes the training way less effective. Furthermore, the most pictures just don't resemble the variety of situations we want to be able to detect facial landmarks in. The pictures show the faces from a frontal perspective without any tilt or rotation. The rotation or roll can be easily dealt with by using data augmentations but to imitate tilt/yaw is impossible. Not only this, for an accessible model to everyone the labeled images also lack diversity with only a few representations of BIPoC. The amount of good data to train on is the most important part of a CNN model like ours.

Another point to consider is the prediction time of the model and the speed of the Viola-Jones algorithm. Currently the model prediction on a given input takes about 20 milliseconds. For a 30 fps video that works out great but for a 60 fps video there is only $\frac{1000\text{ms}}{60} \approx 17\text{ms}$ between every frame. If one wanted to make a prediction on every frame this becomes an important limitation. For most use cases a detection on every 5th, 10th or even 20th is also fine. The same problem also applies to the Viola-Jones algorithm. We had to limit the usage of it to every 10th frame.

5.2 Possible Improvements

As the saying goes, there is always room for possible improvements and scope for further research. We know that in almost every machine learning and deep learning problems, more and better data results in better model performances. For us a big difference would switching to or including other and better datasets make and also training on actual labeled videos.

5.3 Carbon Emission

All the training was conducted using Nvidia Telsa T4 GPUs (TDP of 70W) from the Google Cloud Platform over Google Colab in the region us-east4, which has a carbon efficiency of 0.37 $\text{kgCO}_2\text{eq/kWh}$. For a cumulative of about 8 hours of computation the total emissions are estimated to be 0.21 kgCO_2eq of which 100 percent were directly offset by the cloud provider. Estimations were conducted using the Machine Learning Impact calculator [21].

6 Conclusion

For the purpose of detecting facial points, we suggested an efficient Convolutional Neural Network (CNN) model. Our experiments demonstrate that deep neural networks for computer vision issues like facial landmark detection can be integrated with other techniques and algorithms to be applied to video inputs. The evaluation shows that our model is robust in detecting facial landmarks. Despite the limitations, our model performs well with a decent accuracy, not only on the static image but also on the video stream. Additionally, data augmentations, output masking and pre-processing the video feed helps to overcome the limitations of a rather straitened dataset and achieve high accuracy.

Acknowledgement

We would like to express our gratitude to our project supervisor Noel Danz, for providing his guidance and insights. We also want to thank Prof. Dr. Christoph Lippert and the teaching team, Dept. of Digital Health-Machine Learning, Hasso Plattner Institut (HPI), for offering the course Deep Learning in which context this project took place.

Reference

[1] Brian Amberg and Thomas Vetter. “Optimal landmark detection using shape models and branch and bound.” In: IEEE. 2011, pp. 455–462.

[2] Peter N Belhumeur et al. “Localizing parts of faces using a consensus of exemplars.” In: 35.12 (2013), pp. 2930–2940.

[3] Xavier P Burgos-Artizzu, Pietro Perona, and Piotr Dollár. “Robust face landmark estimation under occlusion.” In: 2013, pp. 1513–1520.

[4] Alexander Buslaev et al. “Albumentations: Fast and Flexible Image Augmentations.” In: Information 11.2 (2020). ISSN: 2078-2489. DOI: 10.3390/info11020125.

[5] Xudong Cao et al. “Face alignment by explicit shape regression.” In: 107.2 (2014), pp. 177–190.

[6] Francois Chollet. Deep learning with Python. Simon and Schuster, 2021.

[7] Tim F Cootes et al. “Robust and accurate shape model fitting using random forest regression voting.” In: Springer. 2012, pp. 278–291.

[8] Timothy F Cootes, Gareth J Edwards, and Christopher J Taylor. “Active appearance models.” In: Springer. 1998, pp. 484–498.

[9] Timothy F Cootes et al. “Active shape models-their training and application.” In: 61.1 (1995), pp. 38–59.

[10] Matthias Dantone et al. “Real-time facial feature detection using conditional regression forests.” In: IEEE. 2012, pp. 2578–2585.

[11] Different submissions to the kaggle facial keypoints detection competition. kaggle/University of Montreal. URL: https://www.kaggle.com/competitions/facial-keypoints-detection/code.

[12] Provided by Dr. Yoshua Bengio. Dataset - Facial Keypoints Detection. kaggle/University of Montreal. URL: https://www.kaggle.com/c/facial-keypoints-detection/.

[13] Stefan Duffner and Christophe Garcia. “A connexionist approach for robust and precise facial feature detection in complex scenes.” In: IEEE. 2005, pp. 316–321.

[14] AD Egorov, AN Shtanko, and PE Minin. “Selection of Viola–Jones algorithm parameters for specific conditions.” In: 42.8 (2015), pp. 244–248.

[15] Yoav Freund and Robert E Schapire. “A decision-theoretic generalization of on-line learning and an application to boosting.” In: 55.1 (1997), pp. 119–139.

[16] S.Zafeiriou G.S.Chrysos E.Antonakos and P.Snape. “Offline deformable face tracking in arbitrary videos.” In: IEEE International Conference on Computer Vision Workshops (ICCVW) (2015).

[17] J.Shen S.Zafeiriou G.S.Chrysos J.Kossaifi G.Tzimiropoulos and M.Pantic. “The first facial landmark tracking in-the-wild challenge: Benchmark and results.” In: IEEE International Conference on Computer Vision Workshops (ICCVW) (2015).

[18] Sergey Ioffe and Christian Szegedy. “Batch normalization: Accelerating deep network training by reducing internal covariate shift.” In: International conference on machine learning. PMLR. 2015, pp. 448–456.

[19] Asifullah Khan et al. “A survey of the recent architectures of deep convolutional neural networks.” In: Artificial intelligence review 53.8 (2020), pp. 5455–5516.

[20] Amit Kumar and Rama Chellappa. “Disentangling 3d pose in a dendritic cnn for unconstrained 2d face alignment.” In: 2018, pp. 430–439.

[21] Alexandre Lacoste et al. “Quantifying the Carbon Emissions of Machine Learning.” In: arXiv:1910.09700 (2019).

[22] Lin Liang et al. “Face alignment via component-based discriminative search.” In: Springer. 2008, pp. 72–85.

[23] Saleque, Ahmed Mortuza and Chowdhury, Fahim Shahriar and Khan, Md Rakin Ashraf and Kabir, Rashadul and Haque, AB. “Bengali License Plate Detection using Viola-Jones Algorithm.” In: International Journal of Innovative Technology and Exploring Engineering 9 (2019), pp. 4058–4065.

[24] Juliana Schwarz and Md Tasnimul Hasan. A brief introduction into Convolutional Neural Networks (CNNs). Mar. 2020. URL: https://www.researchgate.net/publication/340117356_A_Brief_Introduction_into_Convolutional_Neural_Networks_CNNs.

[25] Nitish Srivastava et al. “Dropout: a simple way to prevent neural networks from overfitting.” In: The journal of machine learning research 15.1 (2014), pp. 1929–1958.

[26] Yi Sun, Xiaogang Wang, and Xiaoou Tang. “Deep convolutional network cascade for facial point detection.” In: 2013, pp. 3476–3483.

[27] Vincent Tabora. The viola-jones face detection algorithm. Mar. 2020. URL: https://medium.com/0xcode/the-viola-jones-face-detection-algorithm-3eb09055cfc2.

[28] Great Learning Team. Face detection using Viola Jones algorithm. Mar. 2022. URL: https://www.mygreatlearning.com/blog/viola-jones-algorithm/.

[29] G. Tzimiropoulos. “Project-out cascaded regression with an application to face alignment.” In: IEEE Conference on Computer Vision and Pattern Recognition. 2015, pp. 3659–3667.

[30] Michel Valstar et al. “Facial point detection using boosted regression and graph models.” In: IEEE. 2010, pp. 2729–2736.

[31] S Videla and PMA Kumar. “Modified Feature Extraction Using Viola Jones Algorithm.” In: 10 (2018), pp. 528–538.

[32] Paul Viola and Michael J Jones. “Robust real-time face detection.” In: 57.2 (2004), pp. 137–154.

[33] Bing Xu et al. “Empirical evaluation of rectified activations in convolutional network.” In: arXiv preprint arXiv:1505.00853 (2015).

[34] Heng Yang and Ioannis Patras. “Sieving regression forest votes for facial feature detection in the wild.” In: 2013, pp. 1936–1943.

[35] Zhanpeng Zhang et al. “Facial landmark detection by deep multi-task learning.” In: Springer. 2014, pp. 94–108.

[36] Zhanpeng Zhang et al. “Learning deep representation for face alignment with auxiliary attributes.” In: 38.5 (2015), pp. 918–930.

[37] Xiangxin Zhu and Deva Ramanan. “Face detection, pose estimation, and landmark localization in the wild.” In: IEEE. 2012, pp. 2879–2886.

List of Figures