Addressing Computer Vision Challenges using an Active Learning Framework

Christina Tzogka and Ioannis Refanidis

Department of Applied Informatics, University of Macedonia,
156 Egnatia Street, 546 36 Thessaloniki, Greece
{ctzogka, yrefanid}@uom.edu.gr

Abstract. Computer vision has introduced new successful opportunities in everyday life. Recently, there has been a lot of progress, particularly in face recognition and object detection systems. These systems require a large amount of data to be trained with, in order to perform satisfyingly. Active learning addresses this challenge by leveraging a small amount of manually labelled data. This paper builds on state-of-the-art face recognition and object detection models, by implementing optimization techniques that enhance the recognition accuracy. Further training is being introduced by making use of a robust active learning framework that results in creating extended data sets. Finally, the paper proposes an integrated system, which involves efficient techniques of associating face and object identification information, in order to extract (in real-time) as much knowledge as possible from a video streaming.

Keywords: Face Recognition · Object Detection · Active Learning · Deep Learning · Data Set

1 Introduction

Everyday life as well as industry and business functions have been rapidly transformed due to machine learning (ML) applications. Computer vision (CV), as an ML application, has introduced new successful opportunities in healthcare, transportation, banking, security, media monitoring and more. CV systems are trying to imitate the complex structure of the human eye, which can see and understand the environment or a digital image. Although this is a pretty challenging job, there has been progress particularly in face recognition (FR) [1] and object detection (OD) [2] systems. These systems require large amounts of data to be trained with, in order to perform satisfyingly. In most cases, the available data sets include an amount of unlabelled training samples, that is too large to be manually labeled. Active learning (AL) [3] techniques address this challenge by leveraging only a small amount of manually labelled data, in order to train good supervised models. The main idea in an AL approach is that it could perform better than traditional methods, even with a significantly lower amount of training data. An efficient way of labelling a small amount of training data is utilizing user-friendly annotation tools that could annotate a whole video

streaming and export the labelled data in the appropriate format, so that they could fit on systems' demands. This method of constructing data sets could be extremely useful and time-saving, since often it is tough enough or even impossible to find an existing data set that includes the desired classes. Meanwhile, these annotation tools enable a continuous interaction between system and ML engineer, as he can easily decide which classes will enhance the current data set, while watching a video streaming and capturing the desired entities (objects, faces).

The main contribution of this paper is to: a) provide a comprehensive study on the state-of-the-art FR and OD systems implementation and further training, b) propose a robust AL framework of creating extended data sets, c) introduce a sophisticated optimization method that enhances the recognition accuracy, and d) provide efficient techniques of associating FR and OD outputs through the integrated system, in order to extract as much knowledge as possible from an input video, in real-time. The rest of the paper is organized as follows: First we review the state of the art; then we present the employed data set construction techniques; then we present the proposed framework for AL, including face and object recognition association; and finally we identify future challenges.

2 State-of-the-Art Review

The very latest results of FR tests indicate some of the best FR software available on the market. In 2015, Google announced FaceNet [4], which achieved a record accuracy of 99.63% on the Labeled Faces in the Wild (LFW) [5] data set. At the same time OpenFace [6], an open-source implementation inspired by FaceNet, was released. The OpenFace project employs Dlib [7] built, which has an accuracy of 99.38% (on LFW). Dlib employs Deep Learning (DL), while the network architecture is based on ResNet-34 [8].

ImageAI supports state-of-the-art ML models, trained on the ImageNet-1000 [9] data set, for OD and image classification [10]. The most popular OD model supported by ImageAI is YOLOv3 [11] that has been trained on the COCO [12] data set. The convolutional neural network (CNN) within YOLOv3 is inspired by the GoogleNet [13] model. The YOLOv3 model has a 63.7% mean average precision (mAP) score over the 2007 PASCAL VOC [14] data set, while it allows real-time predictions. YOLOv3 is a robust option for OD, since it is faster and certainly more convenient to use, compared to other well-performed models such as Faster R-CNN [15] and Mask R-CNN [16] that are not implementable for real-time.

Dlib and YOLOv3 are open-source projects and highly recommended by literature. Therefore, the approach adopted in this work includes the state-of-the-art Dlib library and YOLOv3 pre-trained model. This decision resulted in real-time predictions with high accuracy.

3 Data Set Construction

The construction of the data set is a crucial step for any ML-related task. The implemented systems introduced in this paper have different needs concerning the training samples.

3.1 Video Annotation Tool

The data set construction phase involved the usage of an open-source video annotation tool, OpenCV-Video-Label, so that both systems requirements could be addressed efficiently. OpenCV-Video-Label tool supports two object tracking algorithms, Re3 [17] and CMT [18]. More precisely, it enables playing a video, defining a bounding box, where a class (face/object) appears and tracking the entity over multiple frames. Each bounding box refers to a class name that is inserted by the user. The tool provides multiple output formats, based on the system's training demands. Our implementation supports two scenarios: i) FR scenario corresponds to the cropped images that include the face regions and ii) OD scenario corresponds to the original images followed by xml/txt files with the precise coordinates of objects' bounding boxes and class names.

3.2 Face Recognition Data

The training process of a FR system requires images of faces, where the face landmarks [19] are visible. To ensure that the accuracy is not reduced due to training samples issues, the following key points have been noted:

- The total number of image samples per class (person) in the data set is suggested to be higher than 50, while every image should contain a unique face.
- The face within the image should be facing the camera, with both eyes visible. Thus, it is preferable to avoid samples with high percentage of hidden face landmarks (e.g., profile view).
- Image resolution is a determining factor in recognition results. The dimensions of the bounding box that encloses the face should be at least 130x130px.

After data collection, further pre-processing (alignment) on training samples is highly recommended by literature. Face alignment [20] includes rotation, scaling, translating, etc, of face landmarks and aims: a) all faces across the entire training set to have approximately identical size (256 x 256 in our case), b) every face to be centered in the image that it appears, and, c) the eyes to lie on the x-axis. The next step is converting the current data set to encodings (that is, 128-d vectors). Dlib's face recognition network is responsible for generating a feature vector per face in the data set. Hence, a 3-D matrix is built, where all the encodings that correspond to the training samples of the FR system are appended. In particular, every face in the data set is being represented by: i) the image file path, ii) the face encodings, and iii) the class name. Once a new face

sample or a completely new class is being inserted in the data set, the matrix is being updated. For the purposes of our experiments, a data set of 10 classes by utilizing the OpenCV-Video-Label annotation tool, as well as freely available video streaming, was constructed. The average number of image samples per class was approximately 65, while the total number of training samples was almost 665.

3.3 Object Detection Data

The training process of an OD system requires objects' images followed by the corresponding files (xml/txt) that include the actual coordinates of each object within the image, as well as the object's class name. The total number of training samples per class is recommended to be higher than 50. The OD accuracy is being enhanced when a variety of object's features (that is, color, size, shape) is being included in the data set and all image samples are representative of their class. As it is already mentioned, the YOLOv3 pre-trained model has been trained on the COCO data set. As a result, it could make predictions for 80 classes. The data set for further custom training consisted of 5 classes (flag, glass, hat, surgery mask, watch). The OpenCV-Video-Label annotation tool, as well as freely available video streaming, was used. The average number of image samples per class was approximately 150, while the total number of training samples was almost 800.

4 The Active Learning Framework

This section introduces the framework for FR and OD, which makes use of AL techniques leading to automated data set enhancement. Our integrated system combines FR and OD models and results in extracting knowledge from both of them in real-time, as well as associating their outputs providing us additional information about frames' contents. Furthermore, a sophisticated optimization method is incorporated into our system, which enhances the recognition accuracy. Meanwhile, a metadata repository is built, that consists of details about every input that undergoes the system (e.g., unique id, duration, frame rate, etc), as well as system variables (e.g., number of classes per model) and output/logs.

4.1 Face Recognition System

Classifier Training: After generating faces encodings, a ML classifier is being trained, in order to make probabilistic predictions for every detected face in the input video. Training is being conducted on encodings column (training samples), in association to the corresponding class name column (labels), which are included in the aforementioned 3-D matrix. According to bibliography, Support Vector Machines (SVM) [21] and K-Nearest Neighbours (KNN) [22] have been proven quite efficient classifiers for encodings classification. A classifier comparison between SVM and KNN is necessary, in order to decide the optimal classifier

in terms of time and accuracy. The implementation of these classifiers relied on the scikit-learn [1] models. For the purpose of classifier comparison $threshold_A$ is defined. Every true and false prediction higher than $threshold_A$ were categorized as true positive (TP) and false positive (FP), respectively. Our experiments were conducted on the initial data set (10 classes). From the very first results, we observed the SVM classifier saving time in both training and recognition process. On the other hand, KNN classifier required further experimentation concerning the appropriate K value, which as it turned out depends on the number of classes. Additionally, we noticed a significant delay during the KNN recognition process for a high number of classes. In particular, we considered the same input video and number of classes, in order to execute two individual recognition processes, one per classifier (that is, KNN and SVM). More precisely, we created a data set of 100 classes consisting of 10 copies of the initial data set. We aimed to estimate only the duration of the recognition process per classifier, therefore duplicates of classes do not reflect on our observations. Ultimately, we noticed that for a large data set (e.g., number of classes ≥ 100) the SVM classifier takes approximately half the time of the KNN classifier. The latter becomes noticeably slower, while the amount of data is being increased, since KNN estimates the distance of every detected face from all the "known" faces in the data set. Hence, we concluded the SVM classifier as appropriate, as it has been proven considerably accurate as well as convenient to use in a FR system, where the predictions should be made in real-time. The proposed SVM classifier's architecture involves linear kernel, while C and gamma parameters are equal to 1.0 and 1/n_features, respectively.

It is worth noting that the number of classes reflects on results. In fact, an extremely low number of classes (e.g., less than 5) could cause an increase of FP predictions, while an extremely high number of classes could result in significant delay during the FR sub-process. For this reason, we copied the initial 10 classes multiple times, in order to create larger data sets of 65, 650 and 6500 classes. Duplicates of classes do not reflect on our observations for the same reason that we described above. Table 1 shows training and recognition time for the same input but different number of classes. The input video is 10 minutes long, while the frame rate has been reduced to 2 frames per second, in order to speedup the process. Usually the original frame rate of a video equals 25 frames per second. However, FR could perform satisfyingly and significantly faster for 2 frames per second. In particular, the reduced frame rate does not affect the recognition accuracy. For the cases of 650 and 6500 classes we attempted to train two individual classifiers instead of one. The two classifiers did not share any class. Taking into account Table 1, we decided that the total number of classes per classifier shouldn't exceed 600, since the training process takes many hours to complete for an extended data set (e.g., 6500 classes). Recognition time is related to the total number of classes of employed classifiers, too. Finally, we noticed that splitting a heavy data set into two or more classifiers addressed the delay issues, for both training and recognition processes.

[1] https://scikit-learn.org/stable/index.html

Table 1. SVM classifier. Comparison of the training and recognition duration for different number of classes and classifiers.

Class Number	Classifier Number	Training Time (sec.)	Recognition Time (sec.)
65	1	2.7	76
650	1	278	146
650	2	3.6	77
6500	1	$6 * 10^4$	977
6500	2	$13 * 10^3$	660

Prediction Filtering: After training one or more SVM classifiers, the system is ready to make predictions on input videos. For every detected face in a video frame, the classifier predicts a probability value (confidence) for each class in the data set, resulting in a matrix where rows and columns correspond to the classes in the current data set and the frame number, respectively. Face identification (per frame) results from the highest confidence value among all classes.

Prediction flickering is a common phenomenon in most FR systems. Flickering renders the inconsistency of face identification through frames, which is really intense when the detected face is moving. Thus, we employed filtering (that is, Algorithm 1), which managed to stabilize the confidence values. In Figure 1 we present the confidence diagram for a TP case. After applying the algorithm, the confidence values have been smoothed, while the flickering phenomenon has been eliminated.

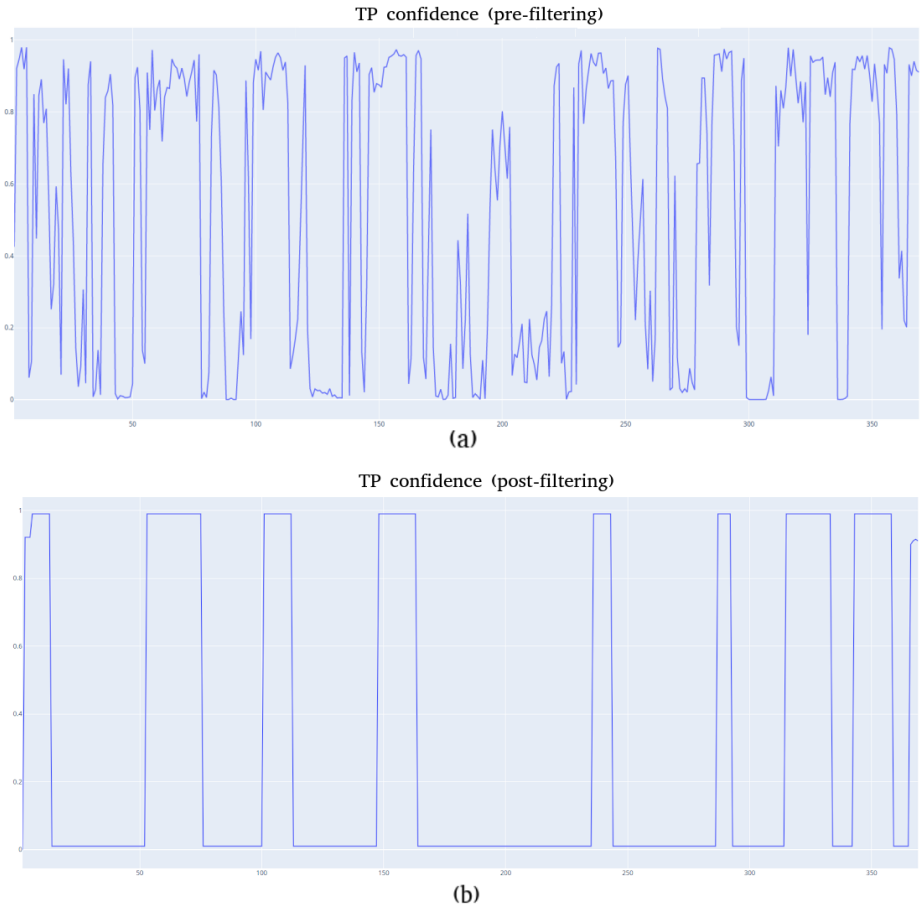
Algorithm 1: FR Prediction Filtering

1. Zero all the confidence values that are under the $threshold_A$, since they probably correspond to FP.
2. Apply a median filter with a N-size kernel. More precisely, re-estimate every class's confidence in a current frame, taking into account the confidences in both previous and next N frames. Then, replace the current value with the median value of the list that has occurred (that is, $V^C list$):

$$V^C[i] = median(V^C_{list})$$

   - $V^C_{list} = \{V^C[i - \frac{N}{2}], V^C[i - (\frac{N}{2} - 1)], ..., V^C[i + \frac{N}{2}]]\}$,
   - V = confidence value per person in the current frame,
   - C = class, i = frame number
3. In the same way, apply a sum filter with a M-size kernel on the updated (from the previous step) confidence values: $V^C[i] = \sum_{i-M}^{i+M} V^C[i]$
4. Finally, replace the updated confidence values as follows:
   - if $V^C[i] > cutoff$ then $V^C[i] = 0.99$
   - else $V^C[i] = 0.01$

Fig. 1. TP example of confidence values through frames, where: (a) represents the pre-filtering confidence values and (b) represents the post-filtering values after applying the Algorithm 1. The confidence values that approach zero in both (a) and (b) correspond to frames, where the current person is out of focus or disappears entirely.



Face Recognition Output: As soon as an input video is being processed for FR, we extract knowledge for both "known" faces, which correspond to data set's classes, and "unknown" faces, which are not included in the data set, yet. Particularly, a face is being classified as "known" if the class's (post-filtering) confidence exceeds 0.99, while a face is being classified as "unknown" if the maximum (pre-filtering) confidence for all classes is under a pre-defined $threshold_B$. When the FR process is over, we export:

- ***(FR-out-1)***: A matrix where rows and columns correspond to the data set's classes and the frame number, respectively. Each cell in the matrix includes

the precise coordinates of the bounding box that encloses the detected face, which appears in the current frame, as well as the confidence score for the corresponding class.

- ***(FR-out-2)***: A dictionary where the keys are the "known" faces that have appeared in the video, while the values per key are: a) the class name, b) a list of the frames where the corresponding class appears, and c) the coordinates of the bounding boxes that enclose the face within the frames, where it appears.
- ***(FR-out-3)***: The encodings of the "unknown" faces that have appeared in the video, in association with the coordinates of the bounding boxes that enclose the faces within the frames, where they appear.
- An annotated video where all the detected faces are within a bounding box, followed by either a class tag or an "unknown" tag.

As a result, we have exploited the information about "unknown" faces, since once a new class is being added in the data set it is possible to detect if it has already appeared in inputs that have undergone FR. For this reason, we estimate the distance between a small set of encodings that have been produced for the new class and those that have been classified as "unknown". If this distance is under a pre-defined $threshold_C$, then both encodings probably represent the same face. Consequently, when an "unknown" class is being inserted in the data set, our AL framework detects all its previous appearances in recent input videos and gives access to more images that could be used as training samples for further training on the current class. In the same way, a "known" class could be enhanced with more samples that originate from predictions with high confidence values. The output video could also be utilized for further training purposes, since it enables the human eye to distinguish the currently "known" from the "unknown" faces. This is undeniably a significant facilitation that results from the AL framework that we have implemented in this paper.

4.2 Object Detection System

Training and Evaluation: ImageAI provides detailed documentation [2] for object training on custom data sets. Firstly, it is recommended to transfer learning from YOLOv3 pre-trained weights to the custom training process. However, the output layer that is responsible for classifying every detected object does not participate in the custom training. As a result, every custom trained model makes predictions only for classes included in the current data set. Throughout our experiments we concluded that the number of classes per custom training should not exceed 10, in order to achieve good results. Training process ends when the validation loss stops decreasing and mAP score is not being significantly increased in comparison to previous epochs. ImageAI enables saving epochs' checkpoints and provides an evaluation method that estimates the mAP score, in order to find the optimal model between checkpoints.

[2] https://imageai.readthedocs.io/en/latest/customdetection/

Object Detection Output: During the prediction process on an input video, the "known" object-classes are being identified only if their predicted probabilities are higher than a pre-defined *tolerance* value. When the OD process is over, we export:

- ***(OD-out-1):*** The precise coordinates of the bounding boxes that enclose the detected objects in each frame. There is also a confidence value per prediction, which represents the probability that the detected object corresponds to each predicted class.
- ***(OD-out-2):*** The total number of instances of the object present in every frame.
- An annotated video where all the detections that have been identified are within a bounding box followed by a class tag.

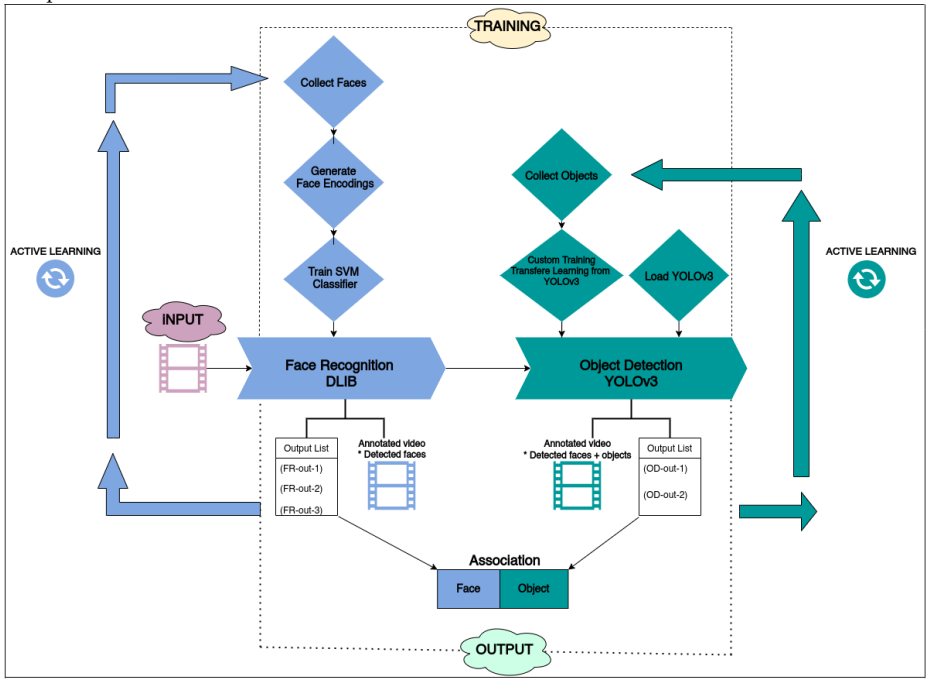
It is worth mentioning that when an input undergoes our OD system multiple times for different custom trained models, the most recent output is being concatenated with the existing (from previous OD executions), in order to maintain the total information. Additionally, we should emphasize the utility of our AL framework, since at the end of the process we collect information about brand-new training samples. A handy scenario is enhancing the data set with the high-confidence samples. Also, the annotated video is extremely useful for data set enhancement purposes.

4.3 The Integrated System

As we have already highlighted, the contribution of this paper includes the implementation of a unique system (that is, the integrated system) that could apply both FR and OD to an input video, in order to extract as much information as possible in real-time. It is worth mentioning that the frame rate of every input video that undergoes our system is being automatically reduced to 2 frames per second, in order to speedup the process, since the reduced frame rate does not affect the recognition accuracy. Figure 2 represents the architecture of the integrated system. Particularly, there are three basic phases: i) the training phase that involves the data set construction, as well as the classifier training (per model, FR & OD), ii) the detection/recognition phase, where the system exports the final outputs that include the identified information, and iii) the AL phase, where our framework leverages the information that we have extracted from the outputs, in order to enhance the training set.

Furthermore, we succeeded in combining the information that we have extracted from both modules of the integrated system, addressing the challenge of faces and objects association that, to the best of our knowledge, has not been implemented before. Therefore, we built a list of personal objects (that is, wearables, such as mask/tie, or objects that a person could use, such as cellphone/umbrella), and we utilized the bounding boxes of both faces and personal objects that have appeared in overlapping frame regions. The output of such an association could be the following system log: *"Object with name:* $\{X\}$ *belongs to a person with name* $\{Y\}$*"*.

Fig. 2. Architecture of the Integrated System. The system is composed of three individual phases: i) the training phase, ii) the detection/recognition phase and iii.) the AL phase.



5 Future Challenges

There are several challenges for future research. Firstly, the proposed method addresses only face to objects associations. To this end, further associations between either individual faces or individual objects could be incorporated, in order to extract statistics about persons or objects' common appearance in videos.

In addition, we plan to include more features in our system's capabilities, in order to satisfy more use-cases. For instance, we could incorporate a color analysis ML algorithm (e.g., K-Means), which would export a color palette consisting of the dominant colors that prevail all over the video. The output palette would enable us to predict the change of the scenery/background through the video. Moreover, an image prediction model could automatically generate tags for each input's frames. As a result, we would be able to estimate the tag variability through the frames as well as determine a possible correlation that may exist between the frames' contents (that is, persons and objects), the generated tag and the dominant colors.

Acknowledgement

This paper is a result of research conducted within the "MSc in Artificial Intelligence and Data Analytics" of the Department of Applied Informatics of University of Macedonia. The presentation of the paper is funded by the University of Macedonia Research Committee.

References

1. Balaban, S.: Deep learning and face recognition: the state of the art. In: Biometric and Surveillance Technology for Human and Activity Identification XII (2015)
2. Girshick, R., Donahue, J., Darrell, T., Malik, J.: Region-Based Convolutional Networks for Accurate Object Detection and Segmentation. In: Transactions on Pattern Analysis and Machine Intelligence, vol. 38, no. 1, pp. 142–158. IEEE Computer Society, Washington (2016)
3. Sener, O., Savarese, S.: Active Learning for Convolutional Neural Networks: A Core-Set Approach. In: ArXiv.org (2018), https://arxiv.org/abs/1708.00489. Last accessed 13 Dec 2020
4. Schroff, F., Kalenichenko, D., Philbin, J.: FaceNet: A unified embedding for face recognition and clustering. In: International Conference on Computer Vision and Pattern Recognition (CVPR). IEEE Computer Society, Washington (2015)
5. Learned-Miller, E., Huang, G. B., Roychowdhury, A., Li, H., Hua, G.: Labeled Faces in the Wild: A Survey. In: Advances in Face Detection and Facial Image Analysis, pp. 189–248 (2016)
6. Baltrusaitis, T., Robinson, P., Morency, L. P.: OpenFace: An open source facial behavior analysis toolkit. In: International Conference on Applications of Computer Vision (WACV). IEEE Computer Society, Washington (2016)
7. Suwarno, S., Kevin, K.: Analysis of Face Recognition Algorithm: Dlib and OpenCV. In: Journal Of Informatics And Telecommunication Engineering, vol. 4, no. 1, pp. 173–184 (2020)
8. He, K., Zhang, X., Ren, S., Sun, J.: Deep Residual Learning for Image Recognition. In: International Conference on Computer Vision and Pattern Recognition (CVPR). IEEE Computer Society, Washington (2016)
9. Fei-Fei, L., Deng, J., Li, K.: ImageNet: Constructing a large-scale image database. In: Journal of Vision, vol. 9, no. 8, pp. 1037–1037 (2010)
10. Nelson, D.: How Does Image Classification Work?. In: Unite.AI (2020), https://www.unite.ai/how-does-image-classification-work/. Last accessed 16 Dec 2020
11. Hurtik, P., Molek, V., Vlasanek, P.: YOLO-ASC: You Only Look Once And See Contours. In: International Joint Conference on Neural Networks (IJCNN) (2020)
12. Lin, T. Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., Dollar, P., Zitnick, C. L.: Microsoft COCO: Common Objects in Context. In: Microsoft Research (2018), https://www.microsoft.com/en-us/research/publication/microsoft-coco-common-objects-in-context/#!abstract. Last accessed 17 Oct 2018
13. Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., Rabinovich, A.: Going deeper with convolutions. In: International Conference on Computer Vision and Pattern Recognition (CVPR). IEEE Computer Society, Washington (2015)

14. Everingham, M., Gool, L. V., Williams, C. K. I., Winn, J., Zisserman, A.: The Pascal Visual Object Classes (VOC) Challenge. In: International Journal of Computer Vision, vol. 88, no. 2, pp. 303–338 (2009)
15. Gad, A. F.: Faster R-CNN Explained for Object Detection Tasks. In: Paperspace Blog (2020), https://blog.paperspace.com/faster-r-cnn-explained-object-detection/. Last accessed 14 Dec 2020
16. He, K., Gkioxari, G., Dollár, P., Girshick, R.: Mask R-CNN. In: ArXiv.org (2018), https://arxiv.org/abs/1703.06870. Last accessed 14 Dec 2020
17. Gordon, D., Farhadi, A., Fox, D.: Re^3: Real-Time Recurrent Regression Networks for Visual Tracking of Generic Objects. In: Robotics and Automation Letters, vol. 3, no. 2, pp. 788–795. IEEE Computer Society, Washington (2018)
18. Nebehay, G., Pflugfelder, R.: Clustering of static-adaptive correspondences for deformable object tracking. In: International Conference on Computer Vision and Pattern Recognition (CVPR). IEEE Computer Society, Washington (2015)
19. Rosebrock, A.: Facial landmarks with dlib, OpenCV, and Python. In: PyImageSearch (2020), https://www.pyimagesearch.com/2017/04/03/facial-landmarks-dlib-opencv-python/. Last accessed 14 Dec 2020
20. Li, H., Wang, P., Shen, C.: Robust Face Recognition via Accurate Face Alignment and Sparse Representation. In: International Conference on Digital Image Computing: Techniques and Applications (2010)
21. Zhang, S., Qiao, H.: Face recognition with support vector machine. In: International Conference on Robotics, Intelligent Systems and Signal Processing, vol.2, pp. 726–730. IEEE Computer Society, Washington (2003)
22. Setiawan, E., Muttaqin, A.: Implementation of K-Nearest Neighbors Face Recognition on Low-power Processor. In: Telecommunication Computing Electronics and Control (TELKOMNIKA), vol. 13, no. 3, p. 949 (2015)