CUSTOMIZED FACIAL EXPRESSION ANALYSIS IN VIDEO

D. Rüfenacht and A. Shaji

Mobius Labs, Germany

ABSTRACT

Existing computer vision based emotion recognition systems are trained to classify images of faces into a very limited number of emotions. In this work, we take a different approach and train a convolutional neural network that is able to distinguish subtle differences in facial expressions appearing in both individual images and videos.
For this effect, we learn a feature embedding network which maps a facial image into a position in embedding space, such that it is positioned close to similar facial expressions compared with other expressions. We train the feature embedding using triplet loss on the publicly available FEC dataset. The proposed facial expression model is lightweight (4.7M parameters), and obtains a triplet prediction accuracy of 84.5% -- very close to the average human performance of 86.2%.
A key contribution of this work is adapting the model which has been trained on images can work reliably on video content. To this end, we propose a novel automatic face quality assessment, which allows us to filter out faces that are unsuited for expression analysis. We validate the proposed approach on several applications, including searching video content for a specific expression from a single query face, generating face expression statistics of various actors appearing in a video, as well as generating face expression summaries.

INTRODUCTION

Facial expressions are one of the most important forms of non-verbal communication between humans. As Alan Fridlund, a psychology professor at University of California Santa Barbara, puts it: “our faces are ways we direct the trajectory of a social interaction”. While often used interchangeably, it is important to highlight that facial expressions and emotions are not the same thing. *Emotions* are things that we *feel,* and are caused by neurons shooting electrons around pathways inside our brains. *Facial expression*s refer to the positions and motions of the muscles beneath the skin of a *face.* They can be an *indicator* of the emotion we are feeling, but do not always reflect our emotion. For example, we can be smiling but be in an emotional state that is far from happy. Rather than teaching a machine the ability to classify a limited set of “emotions”, in this work we set out to train a system that can distinguish facial expressions instead.

A comprehensive and objective way of describing facial expressions is the Facial Action Coding System (FACS), originally proposed by Ekman and Friesen [1]. FACS specifies 32 atomic facial muscle actions which are called action units (AU), which can be used to

objectively describe any facial expression. Since it takes a trained human expert an hour to score one minute of video, a lot of research efforts went into automating facial expression analysis; see Martinez et al. [2] for a comprehensive survey. Due to the complexity involved in training a system to classify a face into 32 AUs – in particular with respect to collecting a large enough dataset – the most popular automatic approaches still resort to directly classifying faces into six (or seven) basic emotions pioneered by Ekman [3], for which a number of publicly available datasets exist. Li and Deng [4] provide a comprehensive overview over the datasets, along with an extensive study on deep learning based categorical facial expression methods.

Our aim is to be able to distinguish facial expressions at a very fine granularity. While appealing from a theoretical point of view, training a system that *explicitly* models the 32 AUs defined by FACS (and then combines them to facial expressions) is a laborious task to say the least. In this work, we follow an idea proposed by Vemulapalli and Agarwala [5], which propose a Facial Expression Comparison (FEC) framework that directly learns to distinguish facial expressions. This is achieved by training a system using triplets of faces, each annotated with which face is most dissimilar to the other two. To the extent that the dataset contains facial expressions where specific AUs are activated, this provides an *intrinsic* way of learning to distinguish different combinations of AUs without the need to explicitly label the faces.

To summarize, the key contributions of this work are:

- a simplified architecture to analyze facial expressions which produces state-of-the-art triplet prediction accuracy;
- the ability to add custom facial expressions by providing just one “reference face”, without the need for retraining the model.
- the ability to distinguish facial expressions at a much more fine-grained level; for example, rather than just being able to tell that someone is smiling, we can reliably quantify the intensity of the smile.
- a novel automatic face quality assessment to enable accurate and stable expression analysis in videos. It is able to filter out faces that are unsuited for expression analysis, like in frames where compression and resolution artefacts have distorted the quality in face pixels, or when the head pose of a person is not suitable to get accurate expression estimations.

The remainder of this paper is organized as follows. First, we present our Facial Expression Embedding architecture (FEENet), which forms a key ingredient in the proposed custom facial expression analysis framework. Next, we show how the per-image facial expression embeddings can be extended to work on videos. In particular, we present an automatic face quality assessment (FaceQA), which allows to filter out faces that are unsuited for facial expression analysis. We then present a quantitative comparison of the FEENet architecture with state-of-the-art, and qualitatively evaluate the customized facial expression analysis on two applications.

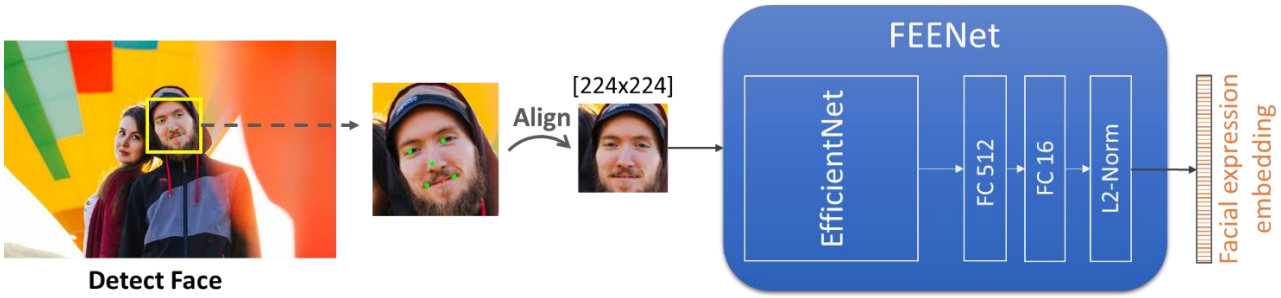

Figure 1. Proposed FEENet architecture to extract facial expression embeddings from images. Each detected face gets aligned into a reference coordinate system and resized. The proposed facial expression embedding network (FEENet) consists of a EfficientNet-B0 backbone, followed by two fully connected (FC) layers that reduce the dimensionality of the embedding vector from 1280 to just 16. Lastly, we apply L2 normalization to obtain the facial expression embedding vector.

LEARNING TO DISTINGUISH FACIAL EXPRESSIONS

Our motivation is to build a method that can distinguish facial expressions at a fine level of granularity which runs efficiently so that it is suited for video analysis. In this section, we give an overview over the key ingredients of how we trained the facial expression embedding network, which allows us to distinguish facial expressions at a fine level of granularity on images. The next section will then show how the framework can be extended to work on videos.

We start by presenting pipeline and architecture used to extract facial expression embedding vectors from images. We then provide some details on the dataset used, and finish the section by providing details about the loss function we use.

Face Expression Embedding Pipeline and Architecture

As shown in Figure 1, the first step of our processing pipeline is to detect the faces in an image; we use the recently proposed RetinaFace by Deng et al. [6]. After this, we use a facial landmark detector to extract the location of the eyes, the nose tip, as well as both sides of the mouth. Much more sophisticated landmark extractors that extract over 60 landmarks exist, but we found this one to be sufficient, as we are only using it to *align* the landmarks into a reference coordinate system; essentially, we undo rotation and scale the face such that the inter-ocular distance (distance between the eyes) is 70px, and crop the face to 224x224 pixels.

This *aligned face* is then input to a CNN, which extracts a feature vector. We tried a variety of CNN architectures, and found that the recently proposed EfficientNet (Tan and Le [7]) performs best as backbone architecture. More specifically, the proposed facial feature extraction uses EfficientNet-B0 as backbone, which outputs a 1280-dimensional feature vector. This feature vector is then passed through two fully-connected layers (FC), reducing the dimensions down to just 16. Lastly, we apply L2-normalization to obtain the facial feature embedding. The resulting facial expression embedding network (*FEENet)* is very simple and lightweight, requiring just 4.7M parameters.

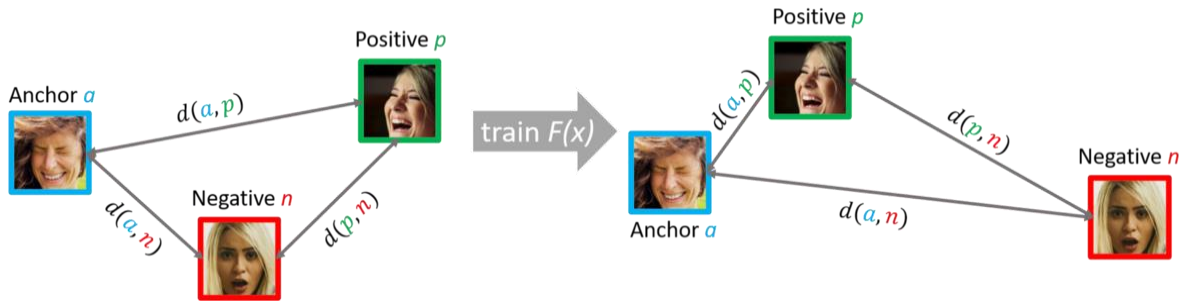

Figure 2. Illustration of the goal of training with triplet loss, which allows to train an embedding where the embedding distance between more visually similar facial expressions is small. $d(i,j)$ represents the distance between facial expressions of face *i* and *j*.

Dataset

In order to train FEENet, we need an appropriate dataset that contains a large number of annotated samples. As mentioned in the introduction, our goal is to learn fine grained details between facial expressions, which cannot be achieved by using datasets that are annotated with a limited set of (basic) emotions. In this work, we use the Facial Expression Comparison (FEC) published by Vemulapalli and Agarwala [5]. In this dataset, we are given triplets of face images, where at least six human annotators had to select the face where the facial expression is most dissimilar to the other two faces of the triplet. The images of these triplets have been carefully sampled from an internal emotion dataset that contains 30 different emotions; see [5] for more details. The original training dataset contains around 130K faces, which are combined to a total of 360K triplets where at least 60% of the raters agreed on the annotation. When we downloaded the dataset, only around 80% was still accessible.

Loss Function

Our goal is to learn a *feature embedding network,* which takes as input a face image and maps it into a location in a facial expression embedding space such that the distance between embedding positions of faces with similar expressions is small, but is large for faces with different expressions. We train the facial feature embedding network using a triplet loss function $L(a,p,n)$ (Chechik et al. [8]), which encourages the distance between the more similar facial expressions (denoted as anchor a and positive p) to be smaller than the distance of these two to the third facial expression of the triplet (denoted as negative n). We can write the triplet loss function as follows:

$$L_{triplet}(a,p,n) = \max\left(0, \|F(a)-F(p)\|_2^2 - \|F(a)-F(n)\|_2^2 + \theta\right) + \max\left(0, \|F(a)-F(p)\|_2^2 - \|F(p)-F(n)\|_2^2 + \theta\right),$$

where $F(x)$ is the feature embedding network, and θ is the *margin* (which we empirically set to 0.2). The margin sets a bound on the minimum distance we want dissimilar facial expressions to have. On the left of Figure 2, we see a possible situation of the distances of one particular triplet (there are hundreds of thousands such triplets in the training set) before training, where the distances between the anchor a, the positive p, and the negative n, are very similar. Using the above triplet loss function, we can achieve that the distance between the anchor and the positive, denoted as $d(a,p)$, is much smaller than $d(a,n)$ and $d(p,n)$.

Figure 3. High-level overview of the proposed facial expression analysis pipeline for videos. First, we run face detection and subsequent automatic face quality assessment (FaceQA), which rejects faces that are blurry and/or too sideways and hence unsuited for facial analysis. Next, we cluster all faces that passed FaceQA into distinct identities. Next, we extract facial expression embeddings using the proposed FEENet, which are temporally pooled for stability.

FACE EXPRESSION ANALYSIS FRAMEWORK FOR VIDEOS

So far, we have shown how we trained a model that is able to distinguish facial expressions. The focus of this work is to propose a practical solution that allows to perform (custom) facial expression analysis in videos. Figure 3 shows the main building blocks of the proposed framework. We run face detection on all video frames, and align them into a common coordinate system. Since not all faces are suited for face expression analysis, we run them through an automatic face quality assessment (FaceQA). All faces that pass FaceQA are then clustered into distinct identities, after which we extract facial expression embeddings using the proposed FEENet. As a last step, the embedding vectors are temporally pooled (per identity) to improve stability and robustness. In the following, we present the fundamental building blocks in more detail.

Automatic Face Quality Assessment (FaceQA)

The face detector and alignment algorithm we use, RetinaFace (Deng et al. [6]), extracts very small and blurry faces also, not all of which lend themselves for subsequent face analysis. We hence propose an automatic FaceQA box that allows us to filter out faces that are unsuited. As shown in Figure 4, we identify two types of faces that cause issues, namely a) blurry faces, and b) faces that are sideways (e.g., facing away from the camera). We now provide a detailed description of the two blocks.

a) Sharpness Score

We compute a sharpness score *sid* to assess the sharpness of an image, which assesses how much high-frequency content is around facial landmarks. In order to avoid that blurry background around the face affects the sharpness score, we assess sharpness only around facial landmarks extracted during face detection.

For all patches, we assess the amount of high-frequency content, obtained by subtracting a blurred patch from the original patch. The sharper the original patch, the larger the difference with respect to the blurred patch will be.

More formally, let us denote the patches of size *MxM* around the landmarks as P_k. Let us further define a Gaussian blur filter as f. Then, the sharpness score for a patch is computed as:

$$sid_k = \frac{1}{M^2} \sum_{i=1}^{M} \sum_{j=1}^{M} |P_k - P_k * f|[i,j]$$

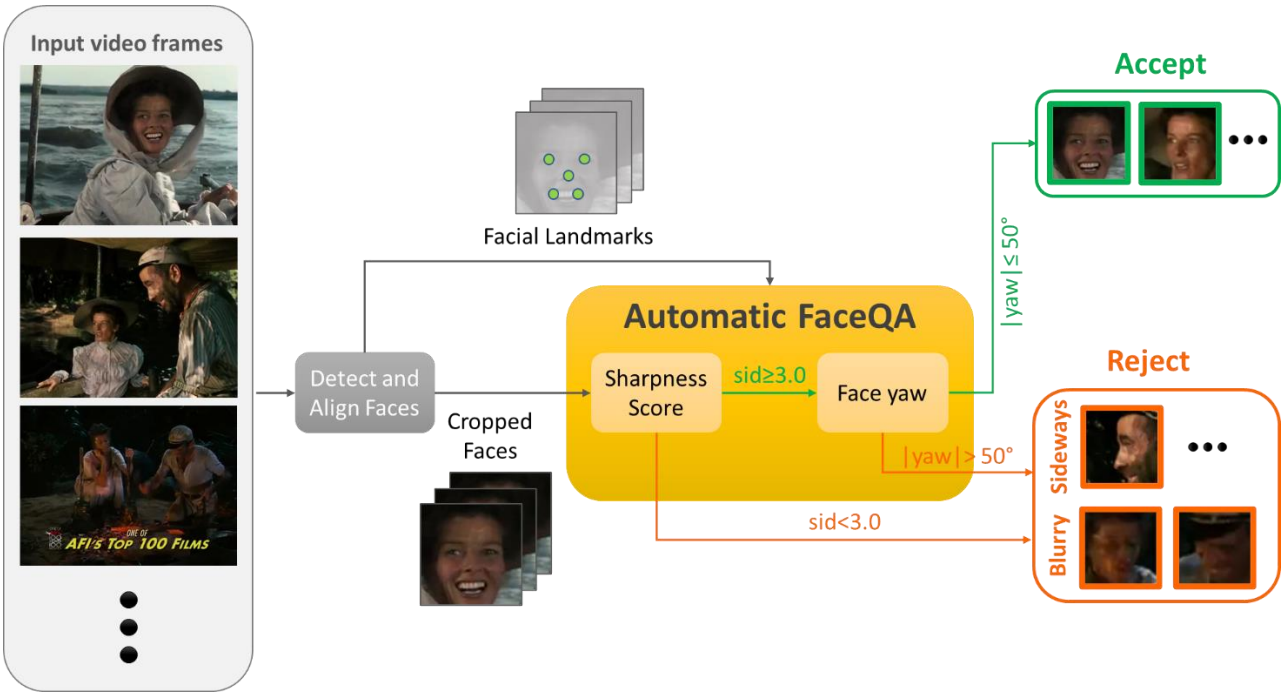

Figure 4. Overview of the proposed automatic face quality assessment. After face detection and alignment, the extracted faces together with facial landmarks are input to FaceQA. First, we compute a sharpness score around facial landmarks. For faces that pass the sharpness threshold, we subsequently compute the face yaw, which allows to reject faces that are too sideways. Only faces that pass both stages are passed onto subsequent facial expression analysis.

The sharpness score is then simply the average of the individual sharpness scores, i.e.,

$$sid = \frac{1}{K}\sum_{k=1}^{K} sid_k$$

In practice, we find that patches around the eyes are most reliable, and hence only assess sharpness there. That is, the sharpness score is simply the average of the sharpness assessed around the two eyes. We empirically set the threshold to 3.0, meaning that we reject all faces that have a sid lower than 3.0.

b) Sideways Faces

In order to get consistent and accurate predictions, we align faces such that the distance between the eyes is 70 pixels. For faces that are too sideways, only one eye is visible and thus the alignment does not work well. We find that such images are frequently causing problems, and hence filter them out.

To this end, we estimate the head pose using perspective-n-point (Lepetit et al. [9]) on the face landmarks extracted during face detection, and reject all faces where

$$|yaw| > 50 \text{ degrees}$$

Faces which are either blurry and/or sideways as defined above are deemed unsuited for further analysis and rejected. All the accepted faces are passed into the next step, which aims at clustering faces into distinct identity clusters.

Automatic Face Clustering into Distinct People Clusters

In order to perform meaningful facial expression analysis on videos, we need to group extracted faces according to their identities. To this end, we use a pre-trained ArcFace-ResNet100 (Deng et al. [10]), which is trained to output face features where the inter-face distance (i.e., face of the same person) is small, whereas the intra-face distance (i.e., face of different but potentially similar looking person) is large. Since the number of distinct identities in a video is unknown, we use a variant of DBSCAN (Ester et al. [11]) to cluster the face features into distinct identities.

This concludes the description of the building blocks that allow us to extend facial expression analysis to videos. In the next section, we first evaluate the facial expression embedding, and then give two practical applications on how the proposed facial expression analysis can be used.

EXPERIMENTS

Face Expression Triplet Prediction Accuracy

This section shows quantitative results for the architecture presented in the *Architecture* section. As mentioned earlier, at the time we downloaded the dataset, only around 80% of the FEC dataset was still available; this fact has to be taken into account when comparing our results with the one from FECNet by Vemulapalli and Agarwala [5], which had access to the whole dataset for training.

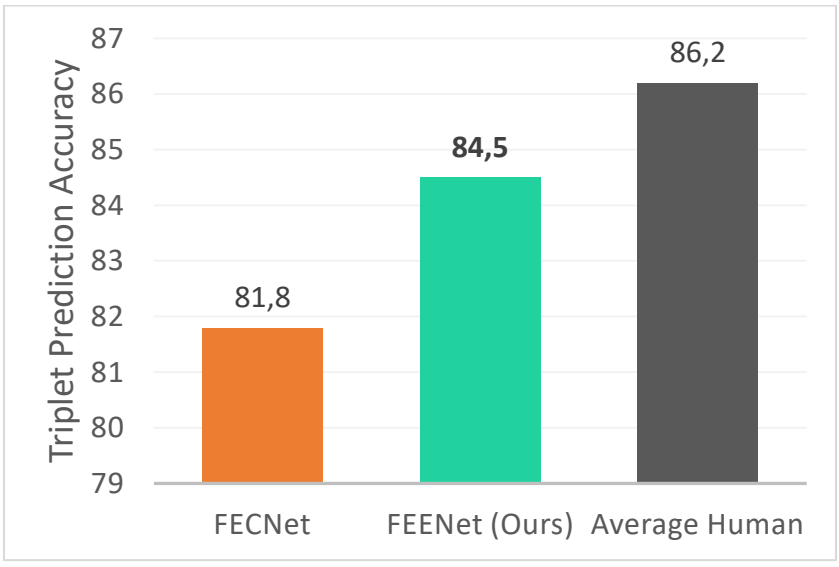

Figure 5. Triplet prediction accuracy for Google AI's FECNet and the proposed FEENet. On the right, we further show the average accuracy of human annotators on the FEC dataset.

Figure 5 compares the triplet prediction accuracy (i.e., the percentage of triplets where the distance between the *anchor* and the *positive* is smallest) of the proposed FEENet with FECNet; in addition, we show the average performance of the *human annotators* who created the ground truth labels for the FEC dataset.

Despite its simpler architecture, the proposed FEENet has an absolute triplet prediction accuracy improvement of 2.7% over FECNet. To put these results in context, it is worth highlighting that the annotators who created the ground truth annotations for the FEC dataset have an average triplet prediction accuracy of 86.2%. In other words, the model we trained almost reaches human performance on this challenging task.

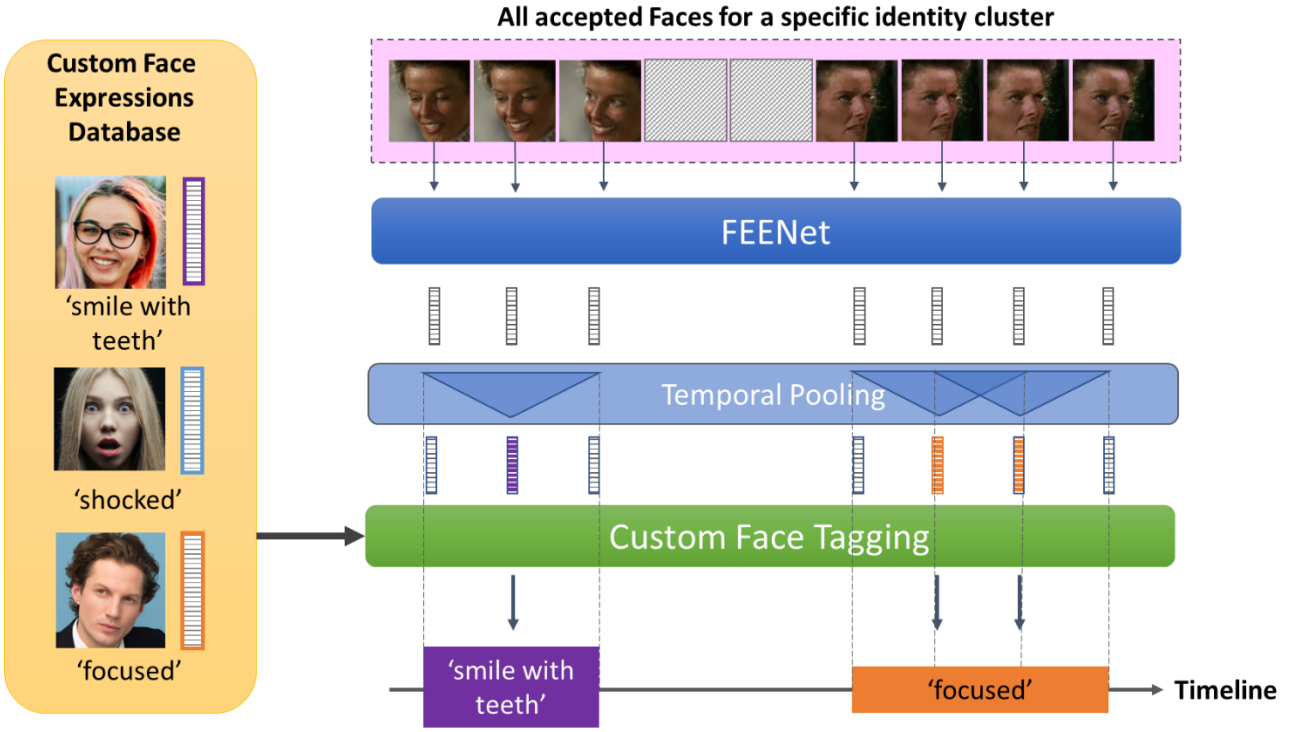

Figure 6. Overview of the proposed custom face tagging framework. Input are all faces that passed the automatic FaceQA, sorted by their temporal order. We then extract facial expression embeddings for all faces, and apply a temporal filter on the embeddings. The pooled embeddings are then compared with the facial expression embeddings extracted from the faces in the custom database, which we show here in different colours for illustrative purposes. If the distance is below a set threshold (0.5 in our case), we mark the corresponding time segment with the matching facial expression.

Application 1: Custom Face Expression Tagging In Videos

We now turn our attention to an interesting application that is enabled by the proposed framework, namely the ability to tag any number of custom facial expressions in videos; we use Figure 6 to guide the description. A key ingredient of the proposed approach is the custom face expression database, which can be populated with faces depicting the facial expressions one seeks to tag. For every face added, we extract the facial expression embedding, together with one or more tags describing the facial expression.

In order to tag the custom facial expressions, we proceed as follows. For every person cluster, we sort the faces according to their timestamps. Next, we use the proposed FEENet to extract the facial expression embeddings. These are then passed through a temporal pooling filter, which allows to control the stability of the facial expression(s). We use simple median pooling with a filter extent of 3 frames, and leave the exploration of more involved pooling strategies for future work. Every pooled facial expression embedding vector is then compared to the custom face expressions database, and if the distance is below a set threshold (0.5 works well in practice), the corresponding time frame gets tagged with the matching custom facial expression.

It is important to highlight that adding new facial expressions does not involve any retraining. All that is required is one face depicting the facial expression of interest, together with the tag(s) describing the facial expression. This is drastically different from existing technologies, where a limited set of predefined emotions are trained (typically less than 10), where it is not possible to add new (emotion) tags without having to retrain the whole architecture using thousands of sample faces that show the desired emotion/facial expression.

Performance in the Wild We choose a recording of the second presidential debate between Donald Trump and Joe Biden[1] as an example to show how the proposed method performs on videos in the wild. As custom face expression database (i.e., reference faces), we extract seven typical facial expressions of Donald Trump, as suggested by a psychologist.[2] We extract all faces for the two identities (Trump and Biden) from the 2-hour long video, and extract facial expression embeddings.

Figure 7 shows the seven ‘reference’ face expressions (in yellow box), as well as the two closest matches to each reference face expression for both Trump and Biden. As can be seen, one reference face is enough for the proposed system to reliably detect the face expressions in the video. As evidenced by the best matches for Biden, face expressions can be queried across identities, allowing for interesting comparisons and statistics on face expressions.

In the figure, we additionally show the appearance rate of specific face expressions, which is defined as the number of times a face expression is tagged, normalized by the total number of faces for an identity. As can be seen, both candidates predominantly exhibited an ‘alpha’ face. While Trump showed an exaggerated mouth in around 20% of the time, Biden smiled a lot more.

[1] https://www.youtube.com/watch?v=bPiofmZGb8o

[2] The seven faces of Donald Trump: https://www.theguardian.com/us-news/2017/jan/15/the-seven-faces-of-donald-trump-a-psychologists-view

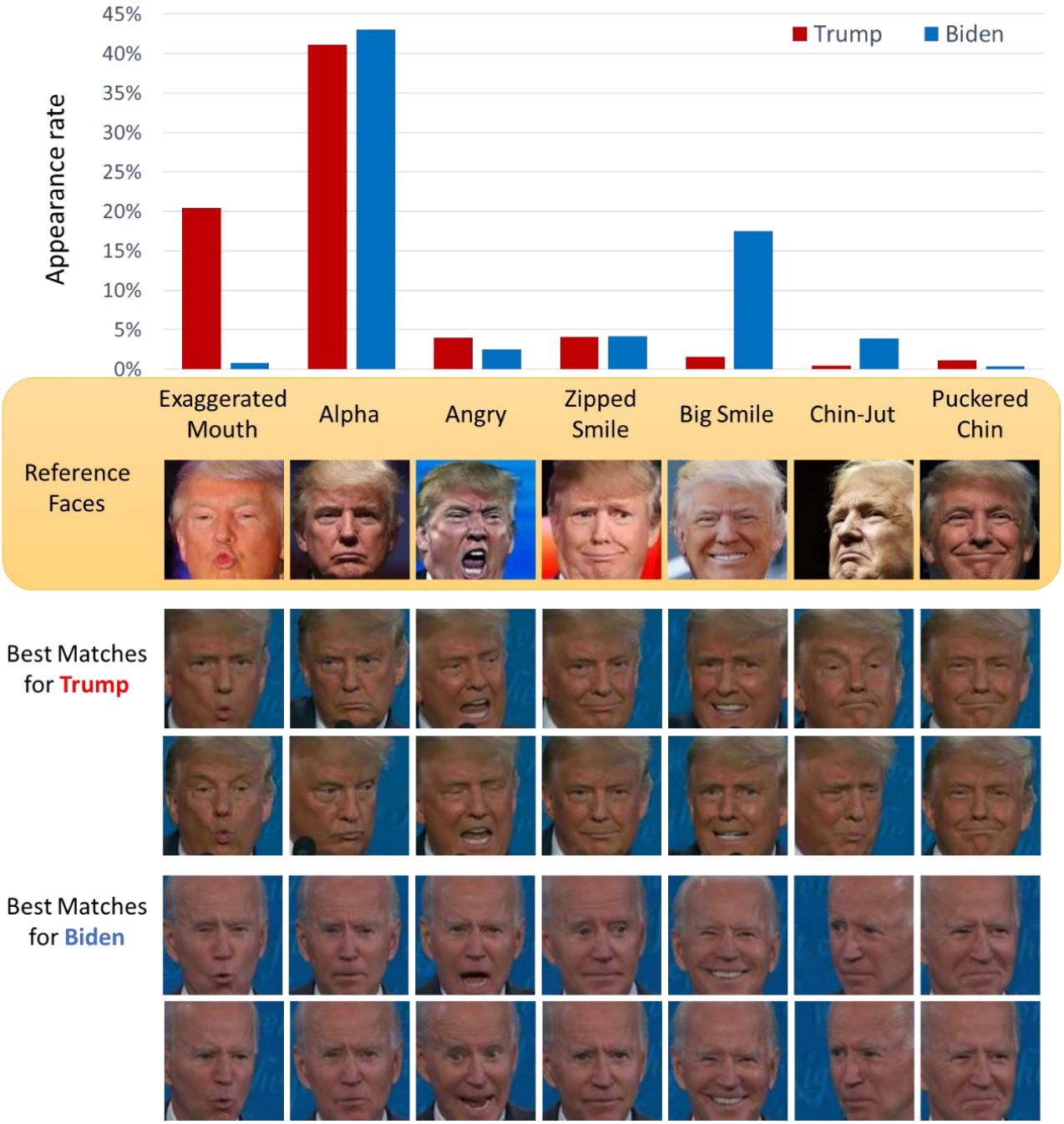

Figure 7. Example of using custom facial expressions to gain meaningful insights into the facial expressions exhibited by different people. Reference faces and names extracted without modifications from article.[2]

Application 2: Facial Expression Summaries

We now turn our attention to another application that the proposed system naturally lends itself to, namely the automatic generation of facial expression summaries. In this setting, rather than looking for similar facial expressions, the aim here is to find dissimilar ones.

Figure 8. Facial expression summary for Jim Carey, with faces sampled from across various movies and TV shows.

Figure 8 shows an example facial expression summary where we extracted faces for Jim Carrey from a range of movies and shows he starred in. We run agglomerative clustering (with N=10 clusters, using ward linkage and Euclidean distance) on facial embeddings extracted from all faces recognized as Jim Carrey, and show one representative face for each cluster. As can be seen, most of the facial expressions shown here are very different from the ones exhibited in the six basic emotions, and hence systems trained on basic emotions would not be able to distinguish them in such detail. Furthermore, the above example also reinforces the importance of being able to search facial expressions via a 'reference' face, as it would be very hard to give a name/tag to most of these faces.

CONCLUSIONS

We have presented a system that can classify custom facial expressions in videos. Unlike prior art that is limited to a small set of emotions, the proposed system can be used to find custom facial expressions without the need of huge amounts of training data. We show that one 'reference face' that portrays the desired facial expression is sufficient to reliably find the facial expression. The current version uses a simple way of pooling face expression features. We demonstrated the capabilities of the proposed face expression features to power two different face perception applications, one for custom face tagging and another for making facial summaries.

The face expression features can be extended to other diverse perception applications, by adding further classification, ranking, or clustering layers on top of it. We are currently exploring applications such as advanced visual similarity search for videos that is able to match and rank actor expressions, creating automatic highlights of interesting segments within and across videos by identifying expressions that tend to interest viewers and many more, which we will report in future work.

REFERENCES

[1] P. Ekman and H. Friesen, Facial Action Coding System: A Technique for the Measurement of Facial Movement, Consulting Psychology Press, 1978.

[2] B. Martinez, M. F. Valstar, B. Jiang and M. Pantic, “Automatic Analysis of Facial Actions: A Survey,” *IEEE Transactions on Affective Computing,* vol. 10, no. 3, pp. 325-347, 2017.

[3] P. Ekman, “Basic Emotions,” *Handbook of Cognition and Emotion,* vol. 98, pp. 45-60, 1999.

[4] S. Li and W. Deng, “Deep facial expression recognition: A survey,” *IEEE Transactions on Affective Computing,* 2020.

[5] R. Vemulapalli and A. Agarwala, “A Compact Embedding for Facial Expression Similarity,” *IEEE Conference on Computer Vision and Pattern Recognition (CVPR),* 2019.

[6] J. Deng, J. Guo, Y. Zhou, J. Yu, I. Kotsia and S. Zafeiriou, “Retinaface: Single-stage Dense Face Localisation in the Wild,” *CoRR,* 2019.

[7] M. Tan and Q. V. Le, “EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks,” *The International Conference on Machine Learning (ICML),* 2019.

[8] G. Chechik, V. Sharma, U. Shalit and S. Bengio, “Large Scale Online Learning of Image Similarity Through Ranking,” *Journal of Machine Learning Research (JMLR),* 2010.

[9] V. Lepetit, F. Moreno-Noguer and P. Fua, “Epnp: An accurate o (n) solution to the pnp problem,” *International journal of computer vision,* vol. 81, no. 2, 2009.

[10] J. Deng, J. Guo, N. Xue and S. Zafeiriou, “Arcface: Additive angular margin loss for deep face recognition,” *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition,* pp. 4690-4699, 2019.

[11] M. Ester, H.-P. Kriegel, J. Sander and X. Xu, “A density-based algorithm for discovering clusters in large spatial databases with noise,” *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining,* pp. 226-231, 1996.