

## SURVEY

# Facial Emotion Recognition for Mobile Devices: A Practical Review

**MICHAL KRUMNIKL**  **AND VOJTECH MAIWALD**

Department of Computer Science, FEEDS, VŠB—Technical University of Ostrava, Ostrava, 708 33 Poruba, Czech Republic

Corresponding author: Michal Krumnikl (michal.krumnikl@vsb.cz)

This work was partially supported by Grant of SGS No. SP2023/072, and SP2024/043 VŠB—Technical University of Ostrava, Czech Republic.

**ABSTRACT** Communicating via email or various chat applications on smartphones is part of most people's daily lives. But in written form, human communication loses a lot of valuable information, such as the facial expressions and emotions of the person you are communicating with. Thanks to techniques from the field of image processing, it is now possible to capture these non-verbal phenomena, and supplement written input with their non-verbal characteristics. In this paper, we explore the possibilities of emotion recognition from front camera images in mobile and embedded devices. A total of 63 classification and 28 regression models based on twelve different neural network architectures optimized for low performance mobile devices were trained and evaluated for success rate and latency. The training and evaluation of each neural network model is performed within the Keras API of the TensorFlow library and then converted to the TensorFlow Lite standard to reduce memory and computational requirements. Great care is taken to ensure that the entire process, from face detection to emotion classification, can operate in real time. To demonstrate and compare the performance of the evaluated models, a freely available optimized application running on Android mobile devices is created and published on Google Play, the source code of which is also available.


**INDEX TERMS** Android, emotion classification, face detection, Keras, MLKit, neural networks, TensorFlow, TensorFlow Lite.

## I. INTRODUCTION

Communication via short messages, email or various chat applications on smartphones is part of most people's daily lives. However, human communication loses a lot of valuable information in its written form, such as the facial expressions and emotions of the person you are communicating with.

The universality of facial expressions of emotion is one of the still debated issues in the biological and social sciences [1], [2], [3], [4], [5], [6]. Darwin's universality hypothesis [7] states that all humans communicate the six basic emotional states (joy, surprise, fear, disgust, anger and sadness) through the same facial movements based on their biological and evolutionary origins.

These non-verbal signals can accentuate the meaning of verbal messages (through accompanying gestures and

The associate editor coordinating the review of this manuscript and approving it for publication was Zhengqing Yun .

grimaces), complement them, but they can also completely change the meaning of what is being communicated (e.g., an ironic grin accompanied by the comment "You did it really well" versus the same sentence spoken with genuine enthusiasm). It also determines much of how we perceive explicit verbal messages – it has been demonstrated that decoding the meaning of a message varies with facial expression [8]. A now-classic study of non-verbal behavior by Mehrabian [9] and later studies with the contribution of Ferris [10] showed that attitude toward a stranger who said "maybe" was approximately 1.5 times more influenced by the stranger's facial expression than by the tone of his voice.

The question we can ask is whether the representation of emotions using emoji [11] in computer-mediated communication is an appropriate form of informing about the speaker's attitude. Studies [11], [12], [13] have shown that there is no indication that computer-mediated communication is less emotional or less personally engaging than

face-to-face communication. On the contrary, emotional communication online and offline is surprisingly similar, and when differences are found, they unexpectedly point to more frequent and explicit communication of emotion in computer-mediated communication [14]. A research [15] aimed at evaluating emotional responses to facial emoji using physiological and self-assessment measures showed that participants' emotional experience was consistent with the emotions expressed by facial emoji. No gender differences were found, and overall the results suggest that emoji are able to elicit particularly pleasant affective states. In exaggeration, we can say that emojis are the modern form of Facial action coding system [16].

Mobile and wearable devices can act as emotion sensors thanks to their integrated cameras and sensors for various physical variables. They have enough power to evaluate and interpret the sensed values as one of the emotional state [17], [18] and encode it as an appropriate emoji icon. Mobile phone applications can take advantage of the combination of facial recognition from a device's camera and various built-in or Bluetooth connected sensors measuring EEG, heart rate, respiration, body temperature and movement [19], [20], [21]. Facial emotion recognition and its presentation in form of emoji can be used to observe the behavior of mobile phone game players [22], drivers in autonomous cars [23], [24], in enhanced chat application [25] or to create an emotion-aware mobile applications for autistic children [21], [26].

Recently published papers [27], [28], [29], [30] focused on facial expression and emotion recognition have presented their results on specialized mobile platforms (often single board computers, typically Raspberry Pi). Although the authors test their solutions on these embedded devices, we hardly ever see real tests on the most common type of mobile devices – mobile phones. Published papers rarely contain results from practical deployment and operation on most common types of phones.

The main purpose of this paper is to address this, review and compare the most widely used methods for facial emotion recognition and test these methods on several types of the common mobile devices/phones (and Raspberry Pi for reference). The presented results evaluate dozens of different emotion recognition methods [31], [32], [33], [34], [35], [36], [37], [38], [39], [40], [41], [42] following the same methodology and compare them with several commonly available mobile phones. The results provide a comprehensive overview of the field, comparing neural network parameters, sizes and latencies on the mobile devices.

## II. NEURAL NETWORKS FOR MOBILE DEVICES

The simplicity of the architecture is essential for the use of neural networks on mobile phones or microcomputers. Neither the amount of available memory nor the amount of computational power can be expected to be comparable

with desktop computers, and GPU acceleration is not the rule either. Over the years, many approaches have been devised, most of which have the following architectures in common. Probably the most fundamental feature is the use of  $1 \times 1$  convolution, introduced in [43]. It is itself relatively computationally inexpensive, but it makes subsequent passage through other convolutional filters and feature extractors many times easier with only a slight reduction in detection quality and accuracy. Another desirable feature of the architecture is the ability to scale the number of trainable parameters by changing the number of layers of the network or its width. This way we can adapt the architecture to the specific performance of the hardware. Typical activation functions are usually different variants of ReLU [44], be it Leaky ReLU [45], ReLU6 [31] or Hard Swish [33], which are semi-linear and simple to compute.

The following architectures were selected for our review because they theoretically allow sufficient scalability for use on mobile devices or because they are designed specifically for such devices. They also represent most well-known architectures; almost all are directly available in the TensorFlow Keras API. We want to achieve latency suitable for real-time use, so we set a latency threshold of 150 milliseconds for image processing by a particular model. Memory requirements do not play such a critical role, but the smaller the model is, the better.

### A. MOBILENET

MobileNet neural network models [31], [32], [33] are renowned solutions for mobile applications and have been designed for the highest efficiency of operations. MobileNet type networks are successfully used in many applications [46], [47], [48] for facial emotion detection. They belong to convolutional neural networks, but convolution is computed in a smart way. The method is called *depthwise separable convolution*, and splits classical convolution into two steps: the depth-wise convolution, and the point-wise convolution [31].

In TensorFlow [49], the MobileNet neural network model can be built very easily through the Keras API by simply calling the appropriate method and specifying a few network parameters such as the input image size, number of color channels, weights, or the number of desired object classes we would like to distinguish [50]. Another important parameter of the model is *alpha*, which can be used to control the number of trainable network parameters. Furthermore, we can choose between different versions of models from the MobileNet family. The specific versions are MobileNetV1 [31] (2017), MobileNetV2 [32] (2018), MobileNetV3Small, and MobileNetV3Large [33] (2019). Google has released the MobileNetEdgeTPU model [51] (2019) with optimization on special TPU computing units. In the mobile version, this chip architecture is used in the Google Pixel 4 series and newer phones for hardware acceleration of neural networks and AI runtime.

MobileNetV2 is based on the principles of MobileNetV1, but additionally extends its architecture to include inverse residue blocks with a convolution function, which allows combining results from activation functions and results that have passed the convolution block filter [32].

The latest version of MobileNetV3 [33] is declaring up to half lower latency compared to the previous generation, while increasing classification accuracy by an order of percent [50] on the Google Pixel 1 phone with ImageNet dataset weights. Compared to the second generation, it brings a number of improvements. It comes with a new block type Squeeze-and-Excitation (SE) that better take into account feature maps based on their channel dependencies. Also, instead of the ReLU activation function, there is a Hard-swish function, which reduces the number of multiply-accumulate operations (MAC) but preserves nonlinearity [52]. It also comes separately in a version for more powerful and weaker target devices, and both versions can additionally be made minimalist or full. With meaningful settings, the MobileNetV3 model can have 2 to 5.4 million trainable parameters.

### B. NASNET

The abbreviation NAS stands for “Neural Architecture Search”, a reference to the way the network in the original implementation adjusts its own architecture based on overall latency, success rate, and dataset size [34]. The way the network is built also changes the way the network is trained – instead of looking for the overall most successful network, the focus is on the best adaptation of a small convolutional cell to a given problem. NASNet has found its way in driver emotion recognition [53], [54], among others.

NASNet was created in three variants, labeled A, B, and C. The resulting architecture is based on the results of optimizing classification accuracy on the ImageNet dataset [55]. Keeping the same number of trainable parameters, the highest success rate on the CIFAR-10 [56] and ImageNet datasets comes out for type A, which is also implemented in TensorFlow. However, instead of being able to scale the architecture using parameters, the TensorFlow developers decided to make the models available only in the smallest and largest configurations and called them NASNetMobile and NASNetLarge. The number of trainable parameters in the mobile version is 5.3 million and in the full version it is 88.9 million parameters.

### C. SQUEEZENET

Another mobile architecture was developed at a time when neural networks were deepening and the primary goal was to increase success rates. SqueezeNet [35] (2016) retained accuracy comparable to AlexNet [57], but using only about one-fiftieth of the parameters. With one and a quarter million trainable parameters, the authors declare a model size of only 0.5 MB, which is on the order of one-hundredth of that of AlexNet. Also, the training speed is many times faster.

SqueezeNet applications for emotion classification focus mainly on a different data source, such as EEG [58], [59], but there are also applications that use facial images [60].

The innovation concerns in particular the so-called *Fire module*, which consists of a squeeze and an expansion layer. In the first layer, a  $1 \times 1 \times K$  convolution is applied to the input of K channels, so that the output of the channel has only one channel. The second layer performs the expansion to the desired number of channels from the total fire block output, using two sets of  $1 \times 1$  and  $3 \times 3$  convolutions with their subsequent concatenation. Fire blocks, in combination with residual connections between them, allow to extract even hard-to-find features from the image.

### D. EFFICIENTNET

EfficientNet [36] is a family of convolutional neural networks designed by a team from Google Brain (2019). The team set out to create a network architecture that is as efficient as possible in its operation, and that can easily adapt its width and depth to a given problem as required. EfficientNet combines key techniques of the MobileNet and SqueezeNet architectures to extract a high number of features of interest from images with a low number of parameters. EfficientNet is successfully used for emotion classification in a video sequence processing library designed for mobile devices [61], [62].

In addition to the original architecture, EfficientNet has also been released in a revised version, EfficientNetV2 [37] (2021), which implements the so-called *adaptive gradient clipping (AGC)* instead of the traditional *batch normalization (BN)*. This technique should provide a much smoother convergence of success rates during training, while being less computationally intensive [37]. However, it cannot be generally claimed that AGC achieves better success rates for the same number of training epochs as BN; the results depend on the specific network.

Because this architecture is designed from the ground up to be highly scalable, the authors have provided eight optimized configurations in the first generation. The smallest, denoted EfficientNetB0, has just over 5 million trainable parameters and size about 20 MB. The smallest second-generation EfficientNet model has about 2 million more parameters and is about 10 MB larger. The largest model, EfficientNetB7, already has almost 67 million parameters and 250 MB in size [36]. In the second generation, this is matched by EfficientNetV2L, which has doubled the number of parameters and size.

### E. SHUFFLENET

Researchers at the University of Hong Kong are behind another innovative neural network called ShuffleNet [38] (2017). The word *shuffle* in this case refers to one of the key layers of the network. In the residual block, after depth convolution (over all channels of the tensor), the channels

of the feature map are swapped before being followed up with a point convolution. The channels are divided into several groups, the groups are swapped, and the groups are transformed back into channels of the original dimension.<sup>1</sup> By swapping the internal channel structure within the residual block, the network learns to perceive the relationships between the channel groups in the feature map, and the number of parameters can be reduced while maintaining the same classification success rate. At the same time, swapping channel groups is a relatively inexpensive operation, making the network traversal less computationally intensive.

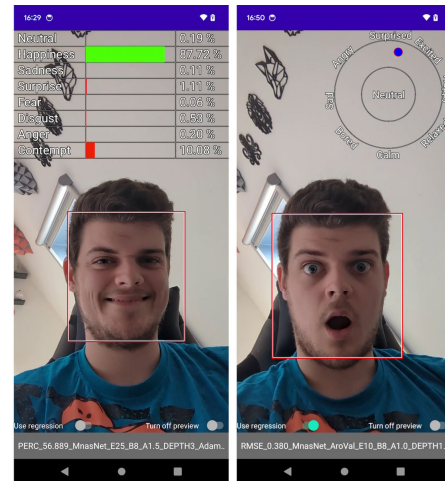
To maximize the success of ShuffleNet detection, it is desirable to divide the channels into a higher number of groups. However, since the convolutions are applied just to the groups, this increases the overall network fragmentation and worsens parallelization (especially on GPUs). So the authors of the original architecture came up with a solution where they included a new type of block called *shuffle block* before each *channel split*, which has two parallel branches, the input tensor is split between them and concatenated at the end. The first branch leaves the input unchanged and the second branch applies the convolutions  $1 \times 1$ ,  $3 \times 3$  and  $1 \times 1$  to it in that order (the input dimension is preserved). In contrast to the convolutions in the *shuffle block*, here they are applied to each channel separately, which makes the parallelization of the necessary computations easier and more efficient, while at the same time the number of *shuffle blocks* to obtain the features can be reduced. This improved architecture has been named ShuffleNetV2 [39] (2018). MiniShuffleNet V2 also occurs in the processing pipeline for realtime face-detection and emotion recognition [63].

### F. DENSENET

DenseNet (2017) is a group of deep neural network architectures that differ precisely in their depth. The version with 121 layers is the shallowest and thus the most suitable for use on a mobile device; other implementations have 169, 201 and 264 layers [40].

DenseNet is often compared to the conceptually similar ResNet [64] architecture, which deals with the problem of vanishing gradient by using residual blocks. However, ResNet will not be discussed in this paper. Initial practical tests have shown that even in the smallest configuration with fifty layers (ResNet50), it is not able to compete with others. Detection accuracy was only average and latency was many times higher for ResNet than for other networks, which is essential for mobile devices. According to these criteria, even the DenseNet121 network cannot be considered a fully mobile architecture. Due to its greater complexity, DenseNet is mainly used for emotion classification with less extensive inputs, e.g., EEG data [65], [66].

<sup>1</sup>Example: From a feature tensor of dimension  $56 \times 56 \times 24$ , the 24 channels are divided into two groups of 12 channels each, resulting in a tensor of dimension  $56 \times 56 \times 2 \times 12$ . The groups swap order and are reshaped and merged back into the dimension of the original channel.



**FIGURE 1. Demonstration application for Android. The left image shows the results of the classification model (MnasNet in this particular case), while the right image shows the outputs from the regression model. The results are displayed at the top of the captured video.**

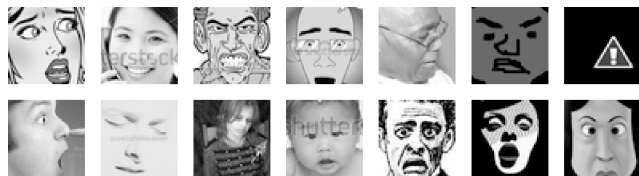
### G. GHOSTNET

The authors of the GhostNet architecture [41] (2019) came up with an interesting finding based on their research into neural network architectures at the time. They found that very often the same or very similar feature maps are generated multiple times from images, which the authors of the architecture found to be inefficient. They proposed a so-called solution *ghost module*, which works by splitting the input data into two parts - the first part is processed by standard convolutions, as in common architectures, but by using fewer parameters. The second part of the module creates other so-called *ghost maps* from these maps in a process that could be likened to augmentation. Using simple operators with linear computational complexity, the feature maps are transformed to produce more, similar maps. Subsequently, the feature maps produced by the convolution pass and the ghost maps are concatenated together into a single tensor. In this way, the authors have streamlined the process of creating very similar symptom maps to each other and thus reducing the computational demands of the network while maintaining comparable success rates. The design of this network is suitable for multimodal emotion recognition [67].

### H. MNASNET

In a similar way to NASNet, the architecture of MnasNet [42] was created, but from the beginning it was designed primarily for use in mobile devices. It takes inspiration from many optimization techniques from other mobile architectures, in particular MobileNet. In designing this architecture, the authors have drawn on Google's insights on feedback learning from the Google AutoML project, which aims to automate the design of machine learning models. In practice, this approach is more applicable to smaller networks and medium sized datasets [68], but automated network designs perform





**FIGURE 2.** Sample of inappropriate images from the FER dataset. It contains a large number of drawings, badly cropped faces, watermark overlays or the face is not even present in the photo. It can also be seen that the original image resolution of only  $48 \times 48$  is really low and many facial details are lost because of this.

well even compared to those designed by human experts. The network is suitable for emotion recognition [69].

The particular form of MnasNet was created using AutoML by continually training and measuring success on automatically generated architectures. In addition to the success rate, the real latency on Pixel phones was also evaluated, and the resulting network design was therefore a compromise of the aforementioned properties of [70]. The resulting architecture has, according to the authors, one-third lower latency than MobileNetV2 and almost two-thirds lower latency than NASNet with comparable success rates. While all of the previous networks mentioned are designed for use on portable and low-power devices, MnasNet is the only network designed primarily for use on phone hardware.

### III. DATASETS

Choosing an appropriate dataset for training the network is an essential step in creating a successful neural network. The ideal dataset should contain as much diverse data as possible, but still stay within the desired categories, and should reflect as closely as possible what the network will then encounter during real deployment. Furthermore, it should be checked that the categorization of the data actually matches its real nature, so that the network is not unnecessarily confused by inconsistencies during training. Ideally, we are looking for a large dataset of photos that match the outputs of the face detectors, i.e., only face cutouts from the chin to the forehead and to the roots of the ears with minimal background.

#### A. FER2013

FER2013 [71] was created based on Google search results and is one of the first ever emotion datasets, which is why it is still widely used for facial emotion recognition. It contains 35,887 black-and-white images of  $48 \times 48$  pixels, each labeled with one of seven emotion categories (anger, disgust, fear, happiness, sadness, surprise, and neutral).

The distribution of categories in the dataset is not very balanced, with almost half of the images with neutral or happy faces combined, and disgust captured in only 547 images. On closer examination, we find that there are also images in the dataset that contain only a face drawing or do not even contain a face (see Figure 2). FER2013 is often criticized for the poor quality of the images and the significant number of

annotation errors. Despite this, success rates on this dataset are still cited in publications as an additional reference.

In 2016, Microsoft released a revision called FER+ [72]. It partially removed the problematic images and completely changed the annotation. There are a total of 12 categories, created by combining the two original ones (happy surprise, sad fear, angry disgust, and so on). This has made it possible to improve the distribution of categories across the dataset, while maintaining the quality of the images.

#### B. RAF-DB

When creating this dataset, the authors focused on selecting photos with the greatest variety in many aspects. The wide range of ages and races of the people captured, as well as the backgrounds, lighting, and quality of the photos, were used to try to approximate real-life conditions as closely as possible. The images were collected from various search engines and social networks and annotated by a group of forty people [73].

The dataset contains about thirty thousand images divided into two groups according to the annotation method. In the first group, the images are divided into seven categories, as is the case for FER2013, and the second group contains twelve categories of combined emotions, as is the case for FER+. The dataset is freely available for non-commercial use on the publication's website and records dozens of citations each year.

#### C. SFEW, EXPW AND CK+

The extended Cohn-Kanade dataset (CK+) is a collection of 593 short video sequences of 123 people of different ages. A facial change from neutral to one of seven other emotions is always captured. The emotions are expressed both purposefully and spontaneously [74]. The representation of race and gender is very diverse, which was also the goal. The main difference from other datasets, however, is the very laboratory/artificial setting of the sampling. Each subject is always facing the camera with a white background behind. It is therefore more of a data collection for mimic muscle analysis. The amount of images that can realistically be used for the purpose of this paper is very small, on the other hand the quality is high.

Similarly to RAF-DB, the datasets *Static Facial Expression in the Wild* (SFEW) and *Expression in the Wild* (ExpW) try to capture the face as realistically as possible. In both cases, however, the images contain a large number of redundant areas that do not contain a face, so to train the neural networks, the face would first need to be found in the images and cut out at a uniform resolution, only after this modification could the datasets be used.

#### D. AFFECTNET

By far the largest and, at the time of writing, by far the most cited facial emotion dataset is AffectNet [75]. Three kinds of annotations are created for each image: membership in one of eight categories of emotion, two values describing the degree

of arousal and valence of a person's emotion (see Russell's Emotion Diagram<sup>2</sup> [76]) and an array of 68 coordinates of points representing points of interest on the face. It contains roughly 440,000 human-annotated images, of which less than 300,000 are available for query; the other 550,000 were annotated by the ResNet neural network learned on the previous set of images with a reported 65 % success rate in categorical detection.

The annotation was performed independently by two people with an overall agreement rate of 60.7 %, which is a relatively low agreement rate, but the categories of no face, uncertain expression, and no emotion are included in this figure. However, since these three categories are not present in the resulting distributed dataset, after excluding them, the agreement rate is almost 66 %. The agreement valence value expressed as the root mean square error is 0.340 and 0.362 for arousal. In addition, note that the square root mean square error does not take into account the sign (the numbers 3 and 5 are as far apart as -1 and 1, but the error in the wrong sign is not negligible in this case). These examples show how subjective human emotions are and that they cannot always be clearly categorized.

The dataset is useful for training both the detection of facial contours in an image, but also the recognition of emotion within fixed categorical boundaries or an emotion diagram. The images are focused exclusively on the face region and at a constant resolution of  $224 \times 224$ . Due to the interplay of all the aforementioned aspects, AffectNet was selected as the default dataset for use in our evaluation and demonstration implementation.

The authors of a companion publication to the dataset also published reference success rates for categorical emotion detection and RMSE for valence and arousal values. For this, AlexNet was used with a weighted loss function highly penalizing the misidentification of the disgust, contempt, and fear classes because they are the least represented in the dataset. In the categorization across the entire dataset, the highest determined success rate was 58 % and the RMSE values for valence and arousal were 0.37 and 0.41, respectively. These numbers roughly correspond to the agreement of the annotators.

#### IV. EVALUATION IMPLEMENTATION

To improve the ability of the neural network to generalize the features of interests, we perform augmentation. After each training epoch, we randomly apply some combination of image modifications to each individual image. Since we plan to use the neural network to classify images from the

<sup>2</sup>Russell's Emotion Diagram - the vertical axis plots the intensity of the emotion (arousal) and the horizontal axis plots the positivity of the emotion (valence). Moving from top to bottom on the Y-axis, we go from maximum arousal to complete calm, and on the X-axis we express negative experiences on the left and positive ones on the right. The basic idea is that although emotions are primarily divided into four quadrants, when their representative points are close together on the diagram, they share many key characteristics. This emotional model has become widely accepted in psychology and beyond.

front camera of the phone, which tends to be of poorer quality, it is logical to add some level of noise or blur to the dataset, change the contrast or saturation level, or slightly rotate the image. Other possible adjustments such as mirroring, skewing or cropping can be applied to avoid overfitting the neural network.

The augmentation of the AffectNet dataset for the purposes of this paper was mediated by the Python library *imgaug* [77]. The operators were set so that after each epoch, one of the available transformations is randomly applied to each frame of the dataset with a probability of 10 %. In the case of blur, *Gaussian Blur* and *Median Blur* were applied with probability 2 %. Each of the transformations can be applied in random order independently of the others, but always at most once per image.

All the above mentioned models were trained for 25 epochs, after each epoch the success of the network was measured and the models that achieved the highest success rate throughout the training are shown in this comparison. It is certainly worth noting the variation in success rates with different batch sizes, depending on the architecture, a batch of 8 or 16 seems to be the best, with a decrease in success rate as the size is increased further. For the MobileNetV2 architecture, batch 16 was also tested, but the success rates are almost identical to batch 32.

In order to maximize the potential of each architecture, it is necessary to adapt its settings to the problem at hand. Many networks have variable widths and depths, and the number of trainable parameters as well as their memory and computational requirements are directly related to this. This work focuses on the use on mobile devices and the number of 10 million parameters proved to be quite limiting during testing; on an average phone, even with good optimization, such a network cannot operate in real time (a latency of up to 150 ms can be considered acceptable). For a better idea, this corresponds to a model size of about 80 MB, which is about 15 MB after conversion to TensorFlow Lite and optimization.

The conversion of the model from TensorFlow to the TensorFlow Lite standard is absolutely necessary to incorporate the neural network into the testing application running on Android. For one thing, there is currently no current TensorFlow API for Java [49], but also TensorFlow models in the standard format are not at all optimized to run on a mobile phone. Converting to TensorFlow Lite is lossy due to its optimization techniques (the converted models will not have the same success rate as the original ones), but with reasonable settings the difference is quite negligible, while the size of the optimized model is fractional. The average size saving of the models used in this text is around 77 %.

Since the original models are acceptably large for use on mobile, there is no need to focus the optimization on size reduction, and since the architectures used are designed to be compact and efficient, there is therefore no need to significantly target latency. For this reason, we chose a default optimization setting at conversion that represents a tradeoff.

In most architectures there is usually a fixed default input image size (most often  $224 \times 224 \times 3$ ), but some are able to work with general size images if no size is specified in the training settings, for example MobileNetV3. However, such a model cannot be directly converted to TensorFlow Lite, since the generic dimension is replaced by dimension equal to 1 with no possibility to override the value. However, if we would like to convert such a model, it can be done indirectly. Using TensorFlow, we create an identical model (this time with the correct input image dimension), load the weights from an existing model, and compile and save the new one. We have verified that the model created in this way has exactly the same success rate as the original one, and the conversion to TensorFlow Lite is then possible.

The application and source codes with all converted and trained models is freely available on Google Play and can be downloaded to evaluate the methods discussed here. The Figure 1 shows samples from the running application demonstrating the classification of emotions of the user's chosen model. Links to the application and the repository are provided in the SUMMARY of the paper.

### V. EVALUATION RESULTS

In total, 63 classification and 28 regression models were trained based on twelve different neural network architectures. The models differed in their internal parameters, where the implementation allowed, and in the training setup. All models were evaluated already during the initial training after each epoch, and always the most successful one was then converted to the TensorFlow Lite format and retested. For demonstration purposes our mobile application includes all the models presented here and gives the reader the opportunity to try the algorithms on their own mobile phone.

The following Tables 1 and 2 show the ten best performing models among the regressors and the twenty among the classifiers. The tables contain the parametric settings used for the network itself as well as the training settings. In the last two columns, the RMSE or the percentage success rate achieved by the model before and after optimization and conversion to TensorFlow Lite is given.

To shorten the descriptions of the network parameters, the following abbreviations are used in the tables: A = Alpha; D = Depth; CH = Channels; SF = Scale Factor; B = Bottleneck; C = Compression; MINI = Minimalistic; LR = Learning Rate; TF = TensorFlow; TFLite = TensorFlow Lite.

By far the most successful is undoubtedly MnasNet, which appears here with different settings several times, closely followed by different versions of ShuffleNet. The differences between the achieved RMSE values are minimal, and therefore all models in the table can be considered pretty much equivalent in terms of their ability to determine emotions. Another interesting fact that can be gleaned from the Table 1 is that no model experienced a change in the average RMSE value on the test dataset due to optimization and conversion to TensorFlow Lite (the values differ negligibly). Figure 4 shows the RMSE values with

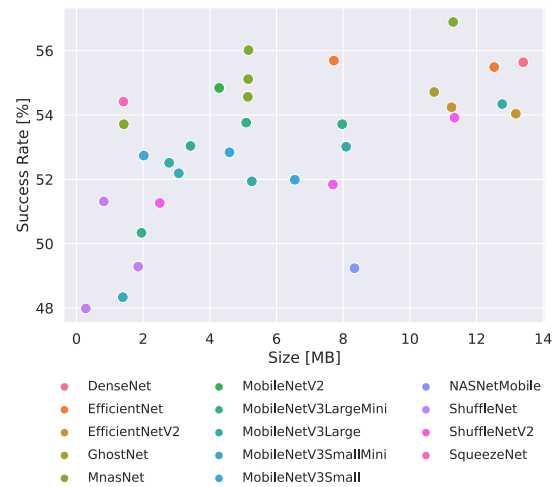


FIGURE 3. Classification models success rate with respect to the size of the models. Intuitively, we can see that the larger the model, the better the classification results.

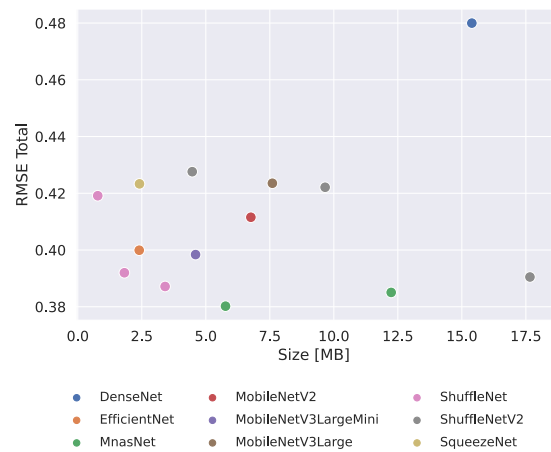


FIGURE 4. Regression models RMSE with respect to the model size. The lower the RMSE, the better a given model is. The contribution of model size to the quality of the results here is not obvious. Despite its size, DenseNet provides the worst results of the entire collection.

respect to the model sizes. As we can see, there is no significant evidence that model size is a key success factor.

Even as a classifier, MnasNet was among the best with a slightly bigger difference than the others in Table 2. ShuffleNet, on the other hand, failed to train to such a success rate and instead EfficientNet is abundantly represented here. As in the previous table, the use of a batch size of 8 appears to be the most optimal for most networks. Further, we see representation of the Adam optimizer with slightly greater frequency, but it cannot be concluded that it is superior. In the case of the classifiers, we can already notice slight differences in success rates after converting to TensorFlow Lite, even for some models the success rate paradoxically increased slightly. In Figure 3 we can see that the success rate of the network essentially increases with the size of the model.

The AlexNet network RMSE reference trained by the authors of the AffectNet dataset was 0.37 for valence and

TABLE 1. Top ten regression models.

Model	Batch	Dropout	Optimizer	LR	RMSE	
					TF	TFLite
MnasNet [42] A = 1.0 D = 1	8	0.2	Adam	0.001	0.380	0.380
MnasNet [42] A = 1.0 D = 1	8	None	Adam	0.001	0.384	0.384
MnasNet [42] A = 1.5 D = 3	8	0.2	SGD	0.01	0.385	0.385
ShuffleNet [38] CH = 200	8	0.2	Adam	0.001	0.387	0.387
ShuffleNetV2 [39] SF = 1.5 B = 1	8	0.2	SGD	0.01	0.390	0.390
MnasNet [42] A = 1.0 D = 1	8	0.5	Adam	0.001	0.391	0.391
ShuffleNet [38] CH = 128	8	0.2	Adam	0.001	0.392	0.392
EfficientNetB0 [36]	8	0.2	SGD	0.01	0.392	0.392
MobileNetV3Large [33] MINI	16	0.2	SGD	0.01	0.398	0.398
MobileNetV2 [32]	8	0.5	Adam	0.01	0.399	0.399

TABLE 2. Top twenty classification models.

Model	Batch	Dropout	Optimizer	LR	Success rate [%]	
					TF	TFLite
MnasNet [42] A = 1.5 D = 3	8	None	Adam	0.000 1	56.864	56.889
EfficientNetB0 [36]	8	None	SGD	0.01	56.164	56.164
MnasNet [42] A = 1.0 D = 7	8	None	Adam	0.000 1	55.989	56.014
EfficientNetB0 [36]	8	None	Adam	0.001	55.914	55.939
EfficientNetB0 [36]	8	None	Adam	0.001	55.714	55.689
DenseNet121 [40]	8	None	Adam	0.000 1	55.639	55.639
EfficientNetB1 [36]	8	None	SGD	0.01	55.564	55.489
EfficientNetB0 [36]	4	None	Adam	0.001	55.264	55.264
MnasNet [42] A = 1.0 D = 3	8	None	Adam	0.000 1	55.139	55.114
EfficientNetB0 [36]	4	None	SGD	0.01	55.089	55.114
MobileNetV2 [32]	8	0.2	Adam	0.001	54.839	54.839
GhostNet [41]	8	None	Adam	0.000 1	54.714	54.714
MnasNet [42] A = 1.0 D = 1	8	None	Adam	0.000 1	54.589	54.564
MobileNetV3Large [33] A = 1.25	16	0.5	Adam	0.001	54.489	54.489
EfficientNetV2B1 [37]	16	None	SGD	0.01	54.464	54.439
MobileNetV3Large [33] A = 1.25	16	0.2	Adam	0.001	54.339	54.339
EfficientNetV2B0 [37]	8	None	SGD	0.01	54.264	54.239
SqueezeNet [35] C = 1.0	8	0.2	Adam	0.000 1	54.139	54.414
EfficientNetV2B1 [37]	8	None	SGD	0.01	54.064	54.039
MobileNetV3Small [33] A = 1.25	16	0.5	Adam	0.001	53.963	53.938

0.41 for arousal. For the best MnasNet model generated in the context of this paper, these values are 0.34 and 0.42, making it a slightly better model on average, but with less computational effort and significantly smaller size compared to AlexNet. The benchmark success rate of AlexNet as a classifier was 58 % in the original AffectNet paper, and this paper achieved the highest success rate of about 57 %, also with the MnasNet model. In both categories, the results obtained are comparable to the benchmarks, but rather than the excellence of the solution itself, we give most weight here to the limitations of the AffectNet dataset annotation, which is very inconsistent, especially in the case of valence and arousal values.

At the time of writing this paper, the highest classification success rate achieved on the AffectNet dataset was 63.03 % [78] with the EfficientNetB2 model, but this is definitely not suitable for use on a mobile device. The best RMSE values on the AffectNet dataset were achieved with the VGG-Face [79] network, specifically the authors report valence and arousal values of 0.356 and 0.327 [80], respectively. Unfortunately, they do not state in their paper

whether these values were achieved within the output of a single model or whether multiple independent models were trained, which seems more likely. Despite this, the valence value achieved by the MnasNet model in this paper is better.

The two tables 3 and 4 list the sizes and measured average latency of TensorFlow Lite models on devices of these specifications:

- Asus ZenFone 5; CPU Qualcomm Snapdragon 636; GPU Adreno 509; RAM 4 GB; Android 9
- Google Pixel 6; CPU Google Tensor 1; GPU Mali-G78 MP20; RAM 8 GB; Android 14 Beta
- Samsung Galaxy A40; CPU Exynos 7904; GPU Mali-G71 MP2; RAM 4 GB; Android 11
- Raspberry Pi 4; CPU Broadcom BCM2711; GPU Broadcom VideoCore VI; RAM 4 GB; Raspbian

However, only models that differ in network architecture settings are listed, as training parameters (batch, number of epochs, optimizer, etc.) do not affect latency or model size. Table 3 lists the values for regression models and the Table 4 for classification models. We chose 150 milliseconds as the reference latency on the least powerful phone, as it



TABLE 3. Latency of TensorFlow Lite regression models on Android phones.

Model	Size [MB]	Device latency [ms]			
		ZenFone 5	Pixel 6	Galaxy A40	Raspberry Pi 4
DenseNet121 [40]	15.39	663	171	751	3298
EfficientNetB0 [36]	2.40	87	29	79	333
MnasNet [42] A = 1.0 D = 1	5.77	104	39	86	461
MnasNet [42] A = 1.5 D = 3	12.24	249	66	227	562
MobileNetV2 [32]	6.76	94	37	106	472
MobileNetV3Large [33] A = 1.0 MINI	4.60	74	28	78	402
MobileNetV3Large [33] A = 1.0	7.60	83	32	91	746
ShuffleNet [38] CH = 64	0.78	76	36	84	74
ShuffleNet [38] CH = 128	1.82	177	70	170	186
ShuffleNet [38] CH = 200	3.41	351	152	341	338
ShuffleNetV2 [39] SF = 0.5	4.47	57	25	63	1257
ShuffleNetV2 [39] SF = 1.0	9.66	164	60	142	1270
ShuffleNetV2 [39] SF = 1.5	17.66	268	74	253	1321
SqueezeNet [35] C = 1.0	2.41	92	38	94	397

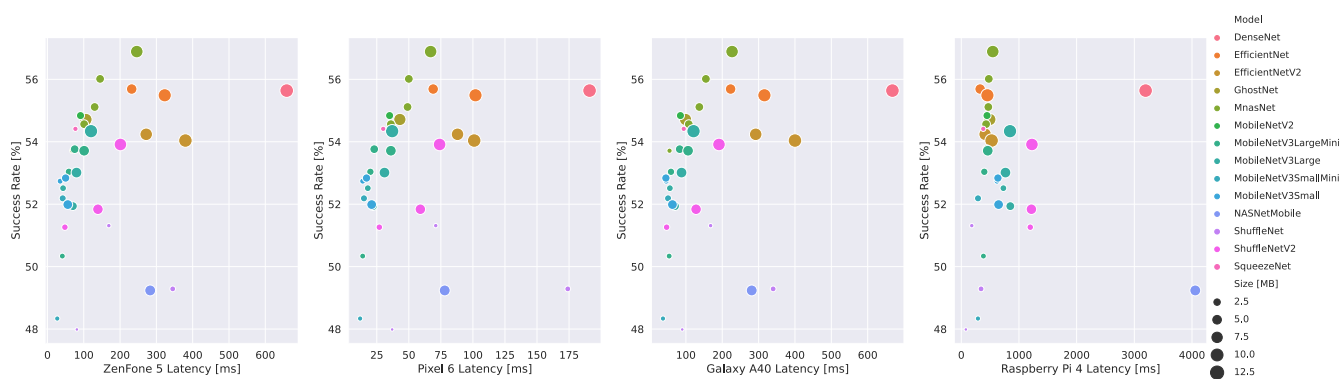
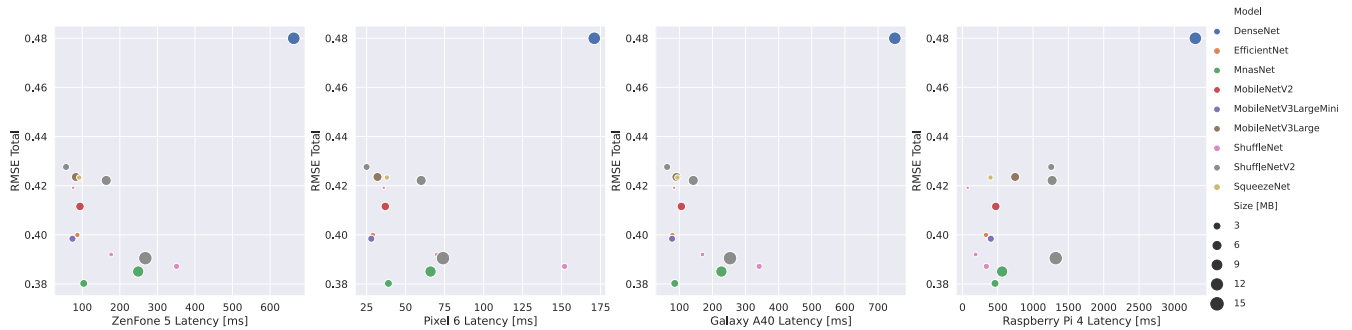


FIGURE 5. Performance of classifier models. Scatter plots showing the relationship between mobile device latency and success rate. Models are shown in different colors, with point size reflecting the size of the model. As we can see, MobileNet networks provide very good performance on all devices while maintaining a small footprint. However, if we consider only latency on powerful devices (like Pixel 6), the difference with MnasNet is negligible.

is not desirable to overload the phone’s system resources unnecessarily when the demonstration application is running. Differences in model sizes relative to the sizes of today’s average applications do not play an important role. All latencies are measured without using TensorFlow Lite delegates, as these are not compatible with some architectures. At the same time, delegates are not supported on all device types, so the most general configuration available on all devices is measured. For faster orientation in the measured data we represented the values of both tables by scatterplots in Figures 5 and 6.

DenseNet121 is the worst in terms of latency and can be excluded from any further comparisons. The same is true for the largest configurations of MnasNet (Alpha = 1.5), ShuffleNet (Channels = 200) and ShuffleNetV2 (Scale Factor = 1.5), which still have too high latency on less powerful devices. The other models can be considered usable, but of course the lower the latency the better. From the Table 1 we know that very good RMSE results were achieved by MnasNet A = 1.0 D = 1, EfficientNetB0 and all MobileNets, so with respect to acceptable latency values we can consider them the best overall.

Even among the classification models, DenseNet121 is again the worst model due to its huge latency and is thus almost unusable for mobile devices. The highest success rate among the classifiers is achieved by MnasNet A = 1.5 D = 3, but it is too demanding, the other MnasNet configurations are nevertheless usable. Unfortunately, the EfficientNet models are a big disappointment even in the smallest configurations; for real-time image analysis on a phone, the latency between 200 and 300 milliseconds is still very high. GhostNet achieved the highest classification success rate of about 54.7 %, which combined with a latency of about 100 milliseconds on a low-performance device is a surprisingly good result. Although the MobileNet architecture models achieve rather average results in emotion classification (MobileNetV2 54.8 %, MobileNetV3Small 53.9 % and MobileNetV3Large 54.5 %), their latency is indeed very low even in the largest configurations. NASNet-Mobile achieves uncompetitive results in both success rate and latency, and there is no point in pursuing it further. The ShuffleNet models in both versions achieve relatively low success rates (around 50 %), and while increasing the network by parameters does increase the success rate, it has too high



**FIGURE 6. Performance of regression models. Graphs showing the characteristics of the models with respect to latency and RMSE values (the smaller the RMSE value, the better the model). The best models are from Mobilenet, EfficientNet and MnasNet groups. We can regard MnasNet as the best because we do not consider the differences in latency to be that significant and MnasNet achieved the lowest RMSE values.**

**TABLE 4. Latency of TensorFlow Lite classification models on Android phones.**

Model	Size [MB]	Device latency [ms]			
		ZenFone 5	Pixel 6	Galaxy A40	Raspberry Pi 4
DenseNet121 [40]	13.40	659	191	668	3196
EfficientNetB0 [36]	7.72	232	69	223	327
EfficientNetB1 [36]	12.53	323	102	316	453
EfficientNetV2B0 [37]	11.25	272	88	292	414
EfficientNetV2B1 [37]	13.18	380	101	400	526
GhostNet [41]	10.73	106	43	99	495
MnasNet [42] A = 0.5 D = 5	1.42	68	23	55	433
MnasNet [42] A = 1.0 D = 1	5.14	101	36	108	430
MnasNet [42] A = 1.0 D = 3	5.15	130	49	137	469
MnasNet [42] A = 1.0 D = 7	5.16	145	50	155	477
MnasNet [42] A = 1.5 D = 3	11.30	246	67	227	546
MobileNetV2 [32]	4.28	91	35	85	446
MobileNetV3Large [33] A = 0.5 MINI	1.95	41	14	54	384
MobileNetV3Large [33] A = 0.5	2.78	43	18	56	730
MobileNetV3Large [33] A = 0.75 MINI	3.42	59	20	59	399
MobileNetV3Large [33] A = 0.75	5.26	70	22	70	850
MobileNetV3Large [33] A = 1.0 MINI	5.09	75	23	83	441
MobileNetV3Large [33] A = 1.0	8.09	80	31	88	766
MobileNetV3Large [33] A = 1.25 MINI	7.97	101	36	106	459
MobileNetV3Large [33] A = 1.25	12.77	120	37	121	845
MobileNetV3Small [33] A = 0.75 MINI	1.39	27	12	37	289
MobileNetV3Small [33] A = 0.75	2.02	35	14	46	619
MobileNetV3Small [33] A = 1.25 MINI	3.07	42	15	51	290
MobileNetV3Small [33] A = 1.25	4.59	50	17	45	635
MobileNetV3Small [33] A = 1.5	6.55	56	21	63	647
NASNetMobile [34]	8.34	283	78	281	4057
ShuffleNet [38] CH = 64	0.28	81	37	90	79
ShuffleNet [38] CH = 128	0.82	169	71	168	182
ShuffleNet [38] CH = 200	1.85	345	174	340	343
ShuffleNetV2 [39] SF = 0.5	2.50	48	27	47	1196
ShuffleNetV2 [39] SF = 1.0	7.69	139	59	128	1216
ShuffleNetV2 [39] SF = 1.25	11.34	201	74	191	1225
SqueezeNet [35] C = 1.0	1.41	77	30	94	381

an impact on latency, and therefore no ShuffleNet model is applicable for real deployment on a mobile device. A pleasant surprise is SqueezeNet, which achieves a success rate of about 54.4 %, has very low phone hardware requirements in all respects, and is also the model with the absolutely shortest training time.

To summarize the above results, the best models with respect to latency and RMSE values achieved are MobileNetV3Large MINI, MobileNetV2, EfficientNetB0 and MnasNet A = 1.0 D = 1. The problem arises more in the general use of regression models to determine valence and

arousal values with training on the AffectNet dataset. This is because the values from the test dataset do not quite match reality.

For the test application, it turns out that it is really not easy to reach a negative value of arousal and thus to be in the lower half on the Russell diagram. Thus, the calmest detectable emotion is only neutral, yet the test dataset contains fairly uniformly distributed values across the entire diagram. Although the detections in the upper half of the diagram are very accurate, we preferred to use a classifier for real deployment.

Among the classifiers, GhostNet, MnasNet A = 1.0 D = 1, SqueezeNet C = 1.0 and MobileNetV2 achieved the highest success rates. The SqueezeNet model excels only in size; if our primary limit were model size, it has no competition, but we consider latency and success rate to be more important. In terms of success rate, the three remaining models are almost identical, which shows that they rank right behind each other in Table 2. However, the difference in latency of over 10 ms for the less powerful devices is in MobileNet's favour. If we consider only latency on powerful devices (like Pixel 6), the difference with MnasNet is negligible. So, also in this case, it very much depends on the specific target application. All three models mentioned above are supported by TensorFlow Lite GPU Delegate, and with such a setup their resulting latency is reduced by up to 65 % on low-performance phones and up to half on Pixel 6.

## VI. SUMMARY

In recent years, we have witnessed a great development of neural networks, and the field of emotion recognition using neural networks has not been left out. This paper offers a summary of approaches related to facial emotion classification, which are widely used in practice, especially in the mobile application segment. We have attempted to demonstrate their capabilities on specific mobile devices commonly encountered and provide a benchmark. The aim of this paper is not to compare the different models in their generality, but on one specific, widely used case - facial emotion classification.

In contrast to individual articles, which are mostly focused on one particular approach, our aim was to compare each method from a practical point of view, which is of most interest to mobile application developers, among others. We focused on success rate, latency and model size, key parameters for practical deployment. The best models with respect to latency and RMSE values achieved in our evaluation were MobileNetV3Large MINI, MobileNetV2, EfficientNetB0 and MnasNet. However, a clear determination of the best model depends on the specific needs of the application; for the expected use, it would probably be MnasNet, since we do not consider the differences in latency to be that significant and MnasNet achieved the lowest RMSE values.

We implemented and trained all models on popular datasets and adapted them to run on Android mobile phones. Based on the feedback from users of our mobile app, we can say that the MobileNetV3 and MnasNet models indeed performed subjectively the best in most of the configurations, which is consistent with their popularity among developers. We also highlight open problems in this area that may inspire new approaches to emotion recognition in mobile phones in the future.

The demonstration application developed for the purpose of this article can be freely downloaded from Google Play <https://play.google.com/store/apps/details?id=cz.vsb.faceemotionrecognition>. The source code is available on

GitHub for readers' convenience, see <https://github.com/VojtaMaiwald/FaceEmotionRecognitionTest>. Please note that the application available on GitHub includes all models, while the application hosted on Google Play doesn't have some of the larger models available due to size restrictions under Google Play policy. No data is transferred to the Internet. All models are stored and run locally on the device, so there is no risk of leaking personal or otherwise sensitive data.

Also, the source code and all the trained models are publicly available, see <https://github.com/VojtaMaiwald/Diploma>. All latency, success rate and RMSE measurements for both standard TensorFlow models and models converted to TensorFlow Lite are also available there. The repository also contains source code for working with the AffectNet dataset, image augmentation, model training and testing, and implementations of neural network architectures not available in the TensorFlow Keras API.

## REFERENCES

- [1] P. Ekman and H. Oster, "Facial expressions of emotion," *Annu. Rev. Psychol.*, vol. 30, no. 1, pp. 527–554, 1979.
- [2] R. E. Jack, O. G. B. Garrod, H. Yu, R. Caldara, and P. G. Schyns, "Facial expressions of emotion are not culturally universal," *Proc. Nat. Acad. Sci. USA*, vol. 109, no. 19, pp. 7241–7244, May 2012.
- [3] J. A. Russell, "Culture and the categorization of emotions," *Psychol. Bull.*, vol. 110, no. 3, pp. 426–450, 1991.
- [4] D. Keltner, J. L. Tracy, D. Sauter, and A. Cowen, "What basic emotion theory really says for the twenty-first century study of emotion," *J. Nonverbal Behav.*, vol. 43, no. 2, pp. 195–201, Jun. 2019.
- [5] R. E. Jack, W. Sun, I. Delis, O. G. Garrod, and P. G. Schyns, "Four not six: Revealing culturally common facial expressions of emotion," *J. Experim. Psychol., Gen.*, vol. 145, no. 6, p. 708, 2016.
- [6] A. Ortony, "Are all 'basic emotions' emotions? A problem for the (basic) emotions construct," *Perspect. Psychol. Sci.*, vol. 17, no. 1, pp. 41–61, 2022.
- [7] C. Darwin, *The Expression of the Emotions in Man and Animals*. London, U.K.: John Marry, 1872.
- [8] M. Spapé, V. Harjunen, I. Ahmed, G. Jacucci, and N. Ravaja, "The semiotics of the message and the messenger: How nonverbal communication affects fairness perception," *Cognit., Affect., Behav. Neurosci.*, vol. 19, no. 5, pp. 1259–1272, Oct. 2019.
- [9] A. Mehrabian, *Silent Messages*, vol. 8, no. 152. Belmont, CA, USA: Wadsworth, 1971, p. 30.
- [10] A. Mehrabian and S. R. Ferris, "Inference of attitudes from nonverbal communication in two channels," *J. Consulting Psychol.*, vol. 31, no. 3, pp. 248–252, 1967.
- [11] Q. Bai, Q. Dan, Z. Mu, and M. Yang, "A systematic review of emoji: Current research and future perspectives," *Frontiers Psychol.*, vol. 10, p. 2221, Oct. 2019.
- [12] S. F. Waterloo, S. E. Baumgartner, J. Peter, and P. M. Valkenburg, "Norms of online expressions of emotion: Comparing Facebook, Twitter, Instagram, and WhatsApp," *New Media Soc.*, vol. 20, no. 5, pp. 1813–1831, May 2018.
- [13] A. Rapp, L. Curti, and A. Boldi, "The human side of human-chatbot interaction: A systematic literature review of ten years of research on text-based chatbots," *Int. J. Hum.-Comput. Stud.*, vol. 151, Jul. 2021, Art. no. 102630.
- [14] D. Derks, A. H. Fischer, and A. E. R. Bos, "The role of emotion in computer-mediated communication: A review," *Comput. Hum. Behav.*, vol. 24, no. 3, pp. 766–785, May 2008.
- [15] C. Gantiva, A. Araujo, K. Castillo, L. Claro, and C. Hurtado-Parrado, "Physiological and affective responses to emoji faces: Effects on facial muscle activity, skin conductance, heart rate, and self-reported affect," *Biol. Psychol.*, vol. 163, Jul. 2021, Art. no. 108142.
- [16] E. Friesen and P. Ekman, "Facial action coding system: A technique for the measurement of facial movement," *Palo Alto*, vol. 3, no. 2, p. 5, 1978.

- [17] J. Shu, M. Chiu, and P. Hui, "Emotion sensing for mobile computing," *IEEE Commun. Mag.*, vol. 57, no. 11, pp. 84–90, Nov. 2019.
- [18] K. Yang, B. Tag, C. Wang, Y. Gu, Z. Sarsenbayeva, T. Dingler, G. Wadley, and J. Goncalves, "Survey on emotion sensing using mobile devices," *IEEE Trans. Affect. Comput.*, vol. 14, no. 4, pp. 2678–2696, Oct./Dec. 2022.
- [19] J. Kim, M. Kang, B. Seo, J. Hong, and S. Kim, "Effective emoticon suggestion technique based on active emotional input using facial expressions and heart rate signals," *Sensors*, vol. 23, no. 9, p. 4460, May 2023.
- [20] O. Piskioulis, K. Tzafilkou, and A. Economides, "Emotion detection through Smartphone's accelerometer and gyroscope sensors," in *Proc. 29th ACM Conf. User Modeling, Adaptation Personalization*, Jun. 2021, pp. 130–137.
- [21] V. Gay and P. Leijdekkers, "Design of emotion-aware mobile apps for autistic children," *Health Technol.*, vol. 4, no. 1, pp. 21–26, May 2014.
- [22] N. Heni and H. Hamam, "Facial emotion detection of smartphone games users," in *Proc. IEEE 28th Can. Conf. Electr. Comput. Eng. (CCECE)*, May 2015, pp. 1243–1247.
- [23] Z. Chen, X. Feng, and S. Zhang, "Emotion detection and face recognition of drivers in autonomous vehicles in IoT platform," *Image Vis. Comput.*, vol. 128, Dec. 2022, Art. no. 104569.
- [24] R. Fusek, E. Sojka, J. Gaura, and J. Halman, "Driver state detection from in-car camera images," in *Proc. Int. Symp. Vis. Comput.* Cham, Switzerland: Springer, 2022, pp. 307–319.
- [25] L. Chong, M. Jin, and Y. He, "EmoChat: Bringing multimodal emotion detection to mobile conversation," in *Proc. 5th Int. Conf. Big Data Comput. Commun. (BIGCOM)*, Aug. 2019, pp. 213–221.
- [26] N. Haber, C. Voss, and D. Wall, "Making emotions transparent: Google glass helps autistic kids understand facial expressions through augmented-reality therapy," *IEEE Spectr.*, vol. 57, no. 4, pp. 46–52, Apr. 2020.
- [27] P. Zhen, H.-B. Chen, Y. Cheng, Z. Ji, B. Liu, and H. Yu, "Fast video facial expression recognition by a deeply tensor-compressed LSTM neural network for mobile devices," *ACM Trans. Internet Things*, vol. 2, no. 4, pp. 1–26, Nov. 2021.
- [28] P. Dhope and M. B. Neelagar, "Real-time emotion recognition from facial expressions using artificial intelligence," in *Proc. 2nd Int. Conf. Artif. Intell. Signal Process. (AISP)*, Feb. 2022, pp. 1–6.
- [29] S. Saxena, S. Tripathi, and T. S. B. Sudarshan, "An intelligent facial expression recognition system with emotion intensity classification," *Cognit. Syst. Res.*, vol. 74, pp. 39–52, Aug. 2022.
- [30] Y. Zhou, W. Zhong, Z. Li, T. Zhang, S. Han, and Q. Shi, "An emotional interaction robot with facial expression recognition realized on raspberry Pi and STM32," in *Advances in Intelligent Automation and Soft Computing*. Cham, Switzerland: Springer, 2022, pp. 1231–1240.
- [31] A. G. Howard, "MobileNets: Efficient convolutional neural networks for mobile vision applications," 2017, *arXiv:1704.04861*.
- [32] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, "MobileNetV2: Inverted residuals and linear bottlenecks," 2018, *arXiv:1801.04381*.
- [33] A. Howard, "Searching for MobileNetV3," 2019, *arXiv:1905.02244*.
- [34] B. Zoph, V. Vasudevan, J. Shlens, and Q. V. Le, "Learning transferable architectures for scalable image recognition," 2017, *arXiv:1707.07012*.
- [35] F. N. Iandola, S. Han, M. W. Moskewicz, K. Ashraf, W. J. Dally, and K. Keutzer, "SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and <0.5MB model size," 2016, *arXiv:1602.07360*.
- [36] M. Tan and Q. V. Le, "EfficientNet: Rethinking model scaling for convolutional neural networks," 2019, *arXiv:1905.11946*.
- [37] M. Tan and Q. V. Le, "EfficientNetV2: Smaller models and faster training," 2021, *arXiv:2104.00298*.
- [38] X. Zhang, X. Zhou, M. Lin, and J. Sun, "ShuffleNet: An extremely efficient convolutional neural network for mobile devices," 2017, *arXiv:1707.01083*.
- [39] N. Ma, X. Zhang, H.-T. Zheng, and J. Sun, "ShuffleNet v2: Practical guidelines for efficient CNN architecture design," 2018, *arXiv:1807.11164*.
- [40] G. Huang, Z. Liu, L. van der Maaten, and K. Q. Weinberger, "Densely connected convolutional networks," 2016, *arXiv:1608.06993*.
- [41] K. Han, Y. Wang, Q. Tian, J. Guo, C. Xu, and C. Xu, "GhostNet: More features from cheap operations," 2019, *arXiv:1911.11907*.
- [42] M. Tan, "MnasNet: Platform-aware neural architecture search for mobile," 2018, *arXiv:1807.11626*.
- [43] M. Lin, Q. Chen, and S. Yan, "Network in network," 2013, *arXiv:1312.4400*.
- [44] A. F. Agarap, "Deep learning using rectified linear units (ReLU)," 2018, *arXiv:1803.08375*.
- [45] A. L. Maas, A. Y. Hannun, and A. Y. Ng, "Rectifier nonlinearities improve neural network acoustic models," in *Proc. ICML*, vol. 30, 2013, p. 3.
- [46] Y. Nan, J. Ju, Q. Hua, H. Zhang, and B. Wang, "A-MobileNet: An approach of facial expression recognition," *Alexandria Eng. J.*, vol. 61, no. 6, pp. 4435–4444, Jun. 2022.
- [47] A. Nouisser, R. Zouari, and M. Kherallah, "Enhanced MobileNet and transfer learning for facial emotion recognition," in *Proc. Int. Arab Conf. Inf. Technol. (ACIT)*, Nov. 2022, pp. 1–5.
- [48] M. K. Chowdary, T. N. Nguyen, and D. J. Hemanth, "Deep learning-based facial emotion recognition for human-computer interaction applications," *Neural Comput. Appl.*, pp. 1–18, Apr. 2021.
- [49] M. Abadi et al. (2015). *TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems*. Software. [Online]. Available: tensorflow.org
- [50] Google. (2019). *TensorFlow-MobileNet*. [Online]. Available: <https://github.com/tensorflow/models/blob/master/research/slim/nets/mobilenet/README.md>
- [51] Google. (2019). *TensorFlow-MobileNetEdgeTPU*. [Online]. Available: <https://github.com/tensorflow/models/tree/master/research/slim/nets/mobilenet>
- [52] M. Šimonik and M. Krumnikl, "Optimized hand pose estimation CrossInfoNet-based architecture for embedded devices," *Mach. Vis. Appl.*, vol. 33, no. 5, p. 78, Sep. 2022.
- [53] D. K. Jain, A. K. Dutta, E. Verdú, S. Alsubai, and A. R. W. Sait, "An automated hyperparameter tuned deep learning model enabled facial emotion recognition for autonomous vehicle drivers," *Image Vis. Comput.*, vol. 133, May 2023, Art. no. 104659.
- [54] K. Zaman, Z. Sun, S. M. Shah, M. Shoaib, L. Pei, and A. Hussain, "Driver emotions recognition based on improved faster R-CNN and neural architectural search network," *Symmetry*, vol. 14, no. 4, p. 687, Mar. 2022.
- [55] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "ImageNet: A large-scale hierarchical image database," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Miami, FL, USA, Jun. 2009, pp. 248–255.
- [56] A. Krizhevsky and G. Hinton, "Learning multiple layers of features from tiny images," Univ. Toronto, Toronto, ON, Canada, Tech. Rep., 2009. [Online]. Available: <https://www.cs.toronto.edu/~kriz/learning-features-2009-TR.pdf>
- [57] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 25, 2012, pp. 1106–1114.
- [58] F. Demir, N. Sobahi, S. Siuly, and A. Sengur, "Exploring deep learning features for automatic classification of human emotion using EEG rhythms," *IEEE Sensors J.*, vol. 21, no. 13, pp. 14923–14930, Jul. 2021.
- [59] Q. Wu, N. Dey, F. Shi, R. G. Crespo, and R. S. Sherratt, "Emotion classification on eye-tracking and electroencephalograph fused signals employing deep gradient neural networks," *Appl. Soft Comput.*, vol. 110, Oct. 2021, Art. no. 107752.
- [60] G. K. Sahoo, S. K. Das, and P. Singh, "Deep learning-based facial emotion recognition for driver healthcare," in *Proc. Nat. Conf. Commun. (NCC)*, May 2022, pp. 154–159.
- [61] A. V. Savchenko, "Video-based frame-level facial analysis of affective behavior on mobile devices using EfficientNets," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. Workshops (CVPRW)*, Jun. 2022, pp. 2358–2365.
- [62] A. V. Savchenko, "HSEmotion: high-speed emotion recognition library," *Softw. Impacts*, vol. 14, Dec. 2022, Art. no. 100433.
- [63] A. Ghofrani, R. M. Toroghi, and S. Ghanbari, "Realtime face-detection and emotion recognition using MTCNN and miniShuffleNet v2," in *Proc. 5th Conf. Knowl. Based Eng. Innov. (KBEI)*, Feb. 2019, pp. 817–821.
- [64] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," 2015, *arXiv:1512.03385*.
- [65] N. Pusarla, A. Singh, and S. Tripathi, "Learning DenseNet features from EEG based spectrograms for subject independent emotion recognition," *Biomed. Signal Process. Control*, vol. 74, Apr. 2022, Art. no. 103485.
- [66] Q. Wu, Y. Yuan, Y. Cheng, and T. Ye, "A novel emotion recognition method based on 1D-DenseNet," *J. Intell. Fuzzy Syst.*, vol. 44, no. 3, pp. 5507–5518, Mar. 2023.
- [67] J. Pan, W. Fang, Z. Zhang, B. Chen, Z. Zhang, and S. Wang, "Multimodal emotion recognition based on facial expressions, speech, and EEG," *IEEE Open J. Eng. Med. Biol.*, early access, Jan. 27, 2023, doi: [10.1109/OJEMB.2023.3240280](https://doi.org/10.1109/OJEMB.2023.3240280).



- [68] B. Zoph, V. Vasudevan, J. Shlens, and Q. V. Le. (2017). *AutoML for Large Scale Image Classification and Object Detection*. [Online]. Available: <https://ai.googleblog.com/2017/11/automl-for-large-scale-image.html>
- [69] S. Aghera, H. Gajera, and S. K. Mitra, "MnasNet based lightweight CNN for facial expression recognition," in *Proc. IEEE Int. Symp. Sustain. Energy, Signal Process. Cyber Secur. (iSSSC)*, Dec. 2020, pp. 1–6.
- [70] M. Tan. (2018). *MnasNet: Towards Automating the Design of Mobile Machine Learning Models*. [Online]. Available: <https://ai.googleblog.com/2018/08/mnasnet-towards-automating-design-of.html>
- [71] I. J. Goodfellow et al., "Challenges in representation learning: A report on three machine learning contests," in *Neural Information Processing*, M. Lee, A. Hirose, Z.-G. Hou, and R. M. Kil, Eds. Berlin, Germany: Springer, 2013, pp. 117–124.
- [72] E. Barsoum, C. Zhang, C. Canton-Ferrer, and Z. Zhang, "Training deep networks for facial expression recognition with crowd-sourced label distribution," 2016, *arXiv:1608.01041*.
- [73] D. Gera and S. Balasubramanian, "Landmark guidance independent spatio-channel attention and complementary context information based facial expression recognition," 2020, *arXiv:2007.10298*.
- [74] P. Lucey, J. F. Cohn, T. Kanade, J. Saragih, Z. Ambadar, and I. Matthews, "The extended Cohn-Kanade dataset (CK+): A complete dataset for action unit and emotion-specified expression," in *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit. Workshops*, Jun. 2010, pp. 94–101.
- [75] A. Mollahosseini, B. Hasani, and M. H. Mahoor, "AffectNet: A database for facial expression, valence, and arousal computing in the wild," *IEEE Trans. Affect. Comput.*, vol. 10, no. 1, pp. 18–31, Jan. 2019.
- [76] J. A. Russell, "A circumplex model of affect," *J. Personality Social Psychol.*, vol. 39, no. 6, pp. 1161–1178, Dec. 1980.
- [77] A. B. Jung. (2023). *Imgaug Library for Image Augmentation in Machine Learning Experiments*. Accessed: Aug. 27, 2023. [Online]. Available: <https://imgaug.readthedocs.io/en/latest/>
- [78] A. V. Savchenko, L. V. Savchenko, and I. Makarov, "Classifying emotions and engagement in online learning based on a single facial expression recognition neural network," *IEEE Trans. Affect. Comput.*, vol. 13, no. 4, pp. 2132–2143, Oct. 2022.
- [79] O. M. Parkhi, A. Vedaldi, and A. Zisserman, "Deep face recognition," in *Proc. Brit. Mach. Vis. Conf.*, 2015, pp. 41.1–41.12.
- [80] A. Bulat, S. Cheng, J. Yang, A. Garbett, E. Sanchez, and G. Tzimiropoulos, "Pre-training strategies and datasets for facial representation learning," 2021, *arXiv:2103.16554*.



**MICHAL KRUMNIKL** received the B.S. and M.S. degrees in psychology from Palack University, Olomouc, and the M.S. degree in information technology with specialization in computer systems and networks and the Ph.D. degree in computer science from the VÁB—Technical University of Ostrava, Czech Republic, in 2006 and 2014, respectively.

Since 2007, he has been an Assistant Professor with the Department of Computer Science, VÁB. He is the author of more than 30 articles. His research interests include image processing, mobile application development, and high-performance computing.



**VOJTĚCH MAIWALD** received the M.S. degree in information technology with specialization in computer graphics from the VÁB—Technical University of Ostrava, Czech Republic, in 2023.

Currently, he is a Developer in an IT company dealing with among other things, the development of mobile applications. His research interests include image processing and application of neural networks.

...