High-performance and Lightweight Real-time Deep Face Emotion Recognition

Justus Schwan, Esam Ghaleb, Enrique Hortal and Stylianos Asteriadis
Department of Data Science and Knowledge Engineering
Maastricht University
Maastricht, Netherlands
justus.schwan@student.maastrichtuniversity.nl, {esam.ghaleb,enrique.hortal,stelios.asteriadis}@maastrichtuniversity.nl

***Abstract*—Deep learning is used for all kinds of tasks which require human-like performance, such as voice and image recognition in smartphones, smart home technology, and self-driving cars. While great advances have been made in the field, results are often not satisfactory when compared to human performance. In the field of facial emotion recognition, especially in the wild, Convolutional Neural Networks (CNN) are employed because of their excellent generalization properties. However, while CNNs can learn a representation for certain object classes, an amount of (annotated) training data roughly proportional to the class's complexity is needed and seldom available. This work describes an advanced pre-processing algorithm for facial images and a transfer learning mechanism, two potential candidates for relaxing this requirement. Using these algorithms, a lightweight face emotion recognition application for Human-Computer Interaction with TurtleBot units was developed.**

I. Introduction

With the recent improvement of the neural network [1], [2], deep architectures have became popular and effective for extracting high level features from data. Recently, deep learning approaches for feature extraction have suppressed the traditional ones and emerged in enormous impact and improvement in many pattern recognition and classification tasks. In computer vision, CNNs are a well-known deep learning architecture for feature extraction from images. In our work, we benefit from this model using the state-of-art VGG-Face representation which proved to be discriminative and efficient at face recognition [2]. While in simpler tasks like handwriting recognition, humans are often outperformed, state of the art algorithms do not yet come close to human performance in more complex fields. One example is facial emotion recognition in the wild. Related work includes person recognition using a CNN [2] and the Facial Expression Recognition Challenge (FER) [3] participants, especially the winner, who employed a CNN with Support Vector Machine (SVM) loss [4].

This work describes the composition of an algorithm achieving state of the art performance by using and fine-tuning existing methods. If an algorithm performs sub-par, this may stem from both the algorithm's design itself, but also from too little or wrong training data. Special focus is therefore put on selection and pre-processing of the data, which seems to be neglected largely in existing work, but can have an immense impact. For the actual recognition task, the CNN trained in [2] was fine-tuned to emotion recognition as a transfer learning approach.

The overall goal of the work is to apply an emotion detection algorithm for HCI purpose in a TurtleBot unit, a small robotics experimentation platform which comes with a low-performance notebook. In addition, we developed Graphical User Interface (GUI), and a client-server application to enable outsourcing of heavy numerical calculations to a remote machine. The CNN will be applicable for more emotion classification tasks in the future.

The rest of work is presented as follow: the second section describes the used datasets, while the third section presents pre-processing pipeline. In the fourth section, we detail the CNN structure and discuss the results. In section five, we introduce the GUI of the real time application. Finally, in section six, we summarize the work and give possible direction for future research.

II. Datasets

When training a neural network, effectively optimizing a high dimensional function, a vast amount of labeled data is needed to overcome the 'curse of dimensionality' [5]. The Acted facial expressions in the wild (AFEW) [6], FER [3] and Cohn-Kanade [7], [8] datasets are publicly available for research purposes and were used in this work.A large share of datasets with emotion labels contain posed facial expressions that were recorded in a controlled environment, with consistent lighting and head pose. Algorithms trained on such a dataset generally perform badly on data with different preconditions, especially in the wild. We expect training on non-similar datasets to yield a better generalization on the performance.

The AFEW dataset consists of 1426 video sequences taken from several movies. Compared to other datasets, which are often recorded in controlled environments, faces in AFEW differ strongly in terms of lighting, position, size and pose. These properties are usually considered undesirable, yet a CNN can learn to abstract meaningful features from data with high intra-class variance. This will potentially affect the performance in real-wold applications positively. Video frames from AFEW contain a lot of redundancy due to the facial expression not varying much between adjacent frames.

978-1-5386-0756-5/17/$31.00

Therefore, the set was sub-sampled to one fifth of its original size.

The FER dataset consists of 28709 training and 7178 validation/test entries. The images were acquired by means of the Google image search API, then labeled and cropped in a semi-supervised way. There is a single face visible on each image, roughly located in the center and facing the camera. After manually verifying the correctness of each crop and label, the entries were converted to grayscale and scaled to 48x48 pixels. Like AFEW, the FER dataset is labeled with the 7 basic emotions: anger, disgust, fear, joy, neutrality, sadness and surprise.

The CK dataset consists of 486 video sequences from 97 posers, CK+ extends the number of sequences by 22%. Data is scaled at either 640x480 or 640x490 pixels and was recorded partially in grayscale. All images were recorded under the same lighting and pose conditions, making the dataset well-suited for use with all methods of feature extraction. The dataset does not contain a default train-test split, and the neutral emotion label is not present in the data.

III. Pre-processing

In data classification tasks, pre-processing is often required and almost always helpful. Our pre-processing pipeline includes face detection and cropping on images which can significantly improve test and training accuracy [9]. While a CNN should, in theory, be able to cope with faces in different locations, lighting conditions and poses, proper pre-processing can reduce intra-class variance and the amount of training data needed. While cropping itself is already a valuable step, the OpenCV Haar Cascade classifier [10] does not yield enough performance for real-time applications, especially on less powerful machines and with rising image resolution. The dlib-ml library [11] provides more sophisticated methods for image detection, object tracking, and facial feature extraction. The latter can be used to further reduce variance between the data by applying an affine transformation to make the positions of eyes and mouth invariant.

A. Real Time Face Detection and Tracking

There are two face detection universal algorithms trained appropriately and readily available in Python. OpenCV provides the Haar Cascade classifier, while dlib-ml uses a HoG-based detector [12]. While the former is employed more frequently, the latter performs faster and more reliable face detection. dlib-ml's detector was trained on 3000 images from the Labeled Faces in the Wild database using an SVM-max margin classifier [13].

For pre-processing the available data, dlib-ml's classifier was primarily used, with OpenCV's only being employed as a fallback when the former would not yield a confident detection. For a 640x480 resolution, image processing times were around 1000ms (OpenCV) vs. 200ms (dlib-ml) with 2x up-scaling. While the performance is absolutely satisfactory for pre-processing a database, even dlib-ml's detection speed is hardly sufficient for real-time applications. For better real-time performance, dlib-ml provides a correlation tracker [14] implementation. It tracks objects in images reliably and performs an iteration on the aforementioned 640x480 pixels image in around 80ms. Once a face has successfully been detected, then the tracker can take over, thus increasing the detection performance to more than ten frames per second and rendering the algorithm suitable for real-time applications.

B. Facial Landmark Detection and Face Alignment

Following face detection, a facial landmark detection algorithms can be applied. One of the most famous and reliable facial landmark detectors is [15] by Kazem et al. In this work, we use an implementation of [15] provided by dlib-ml. The detector has an accurate and reliable performance, providing outlines of eyes, mouth, nose and chin. Using an affine transformation, the location of eyes and mouth can be made invariant while maintaining the facial expression. This method aligns faces cleanly, as can be seen in Fig. 1. Assuming that cropping faces in [9] improved classification performance due to a lower intra-class variance, we expect this pre-processing method to yield even greater improvements by further reducing variance.

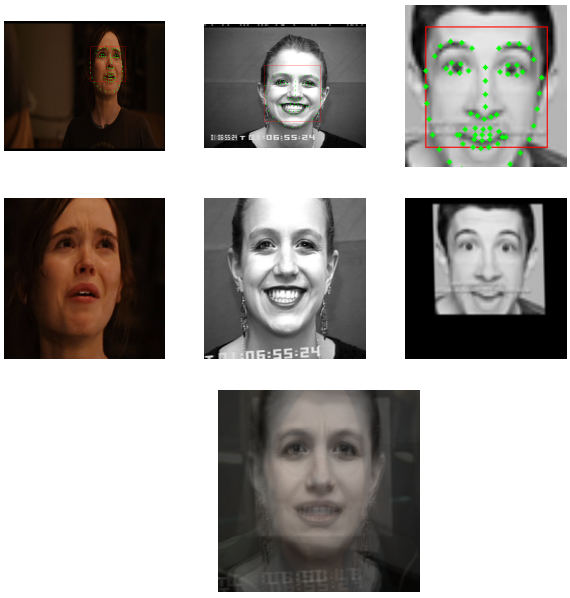


Fig. 1: Samples from AFEW, CK, and FER sequentially. First and second rows show the same images, in original state and with eyes and mouth aligned. Bottom image is a blend of 8 aligned samples to demonstrate alignment precision

IV. Face Representation and Classification

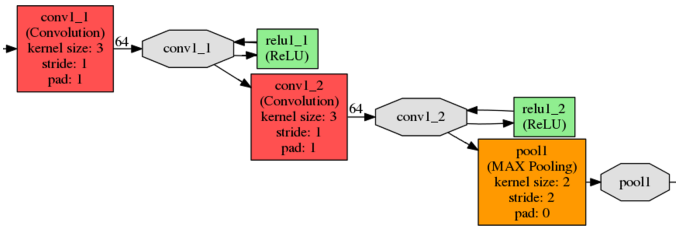
For feature detection and classification, we used a CNN with an attached softmax classifier. Since a conventional CNN for image classification contains millions of parameters and only around 30,000 training samples were available, a transfer learning method was applied to reduce the risk of overfitting. The VGG-Face network [2] is composed of 5 convolutional/pooling and three fully connected layers and was trained for face recognition using 2.6 million images

of 2622 celebrities. The network achieved excellent results (> 90%) for the Labeled Faces in the Wild and YouTube Faces dataset. With such a large amount of face data, VGG-Face's convolutional layers are expected to produce suitable features for faces out of the box. Since the network is freely available in Caffe [16] format for research purposes, we chose to use it as our starting point.

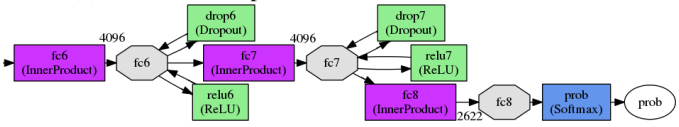
A. Structure of Modified CNN

The convolutional/pooling layers in modern CNNs effectively reduce an m-by-n image to a k-dimensional feature vector, giving a more meaningful description than mere pixel values. While other methods that are based on hand crafted features generate usable features from images, a properly trained CNN is not only specialized on a certain image class but also transforms into a lower-dimensional space. This reduces the required number of training examples drastically due to the 'curse of dimensionality'. The convolutional layers from VGG-Face were taken as-is without any changes.

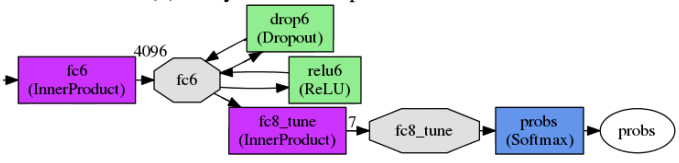
In most networks, two or more fully connected layers follow the convolutional layers, transforming the data to the desired dimensionality. VGG-Face has three fully connected layers in total and yields a 2622-dimensional vector. The last layers are trained on certain celebrities, not on emotions, with each subsequent layer becoming more specialized on this task. We therefore chose to keep only layer 6 for fine-tuning, discard the remaining and append a new 7-dimensional layer with softmax classification and multinomial logistic loss. All fully connected layers in VGG-Face were followed by a dropout unit set to 50%, which we also chose to keep. The network structure of both VGG-Face and our modified version can be seen in Fig. 2.



(a) Convolutional part of VGG-Face, one of five units



(b) Fully connected part of VGG-Face



(c) Fully connected part of our modified network

Fig. 2: The CNN structure

B. CNN Training

To get an impression of the generalization performance, the network was trained and tested on different datasets, as well as combinations thereof. We use an Adam solver [17] as an optimization algorithm with a learning rate of 0.001. For fine-tuning layer 6, the learning rate is reduced to 3%. Training is performed on a Nvidia GeForce GTX TITAN X with 3072 cores in an Intel Xeon E5-2620 2.1Ghz Workstation.

For training and testing, we use the provided train-test splits from AFEW and FER. CK lacks default splits, and its data is only included in the training. The pre-processing algorithm was not able to detect a face in every training/test image, sometimes due to bad visibility and, in the FER dataset, because the image simply did not contain a face. Therefore, we end up with the training/test set sizes depicted in Table I. The training results can be seen in Table II, and the confusion matrix for the seven basic emotions of FER dataset is shown in Fig. 3.

Table III, taken from [3], depicts the FER Challenge winner's performance. The three best contestants used CNNs while the winner outperformed the others with an optimized SVM loss function. The fourth contestant utilized SIFT descriptors for feature learning and a Bag of Words approach for training, almost reaching the third best CNN's performance.

SET	Training	Testing
AFEW	19153	4755
FER	7997	3711
CK	309	
TOTAL	27459	8466

TABLE I: Training and test set sizes

		Tested on		
		AFEW	FER	ALL
Trained on	AFEW	32.6%		
	FER	32.7%	69.8%	
	ALL	31.0%		53.2%

TABLE II: Training Results

Members	Accuracy
Yichuan Tang	71.162%
Yingbo Zhou, Chetan Ramaiah	69.267%
Maxim Milakov	68.821%
Radu Ionescu, Marius Popescu, Cristian Grozea	67.484%

TABLE III: Top Contestants from the FER challenge, as in ([3])

In conclusion, the training results did not vary much from what was to be expected. The AFEW dataset contains relatively few entries, causing severe overfitting and bad test results, with a mere 32.6% test accuracy, even with a heavily increased 90% dropout rate. The FER dataset contains roughly five times as much data, when accounting for redundancy in the former. Results on the dataset reached almost 70%, which is on par with the FER Challenge's contestants.

It has to be noted that the VGG-Face network was originally trained with color data and an image size of 224x224 pixels.

In contrast, FER contains grayscale images, 48x48 pixels in size. Despite the much lower resolution and the missing color information, the network managed to detect very good features, which impressively demonstrates CNN's generalization ability.

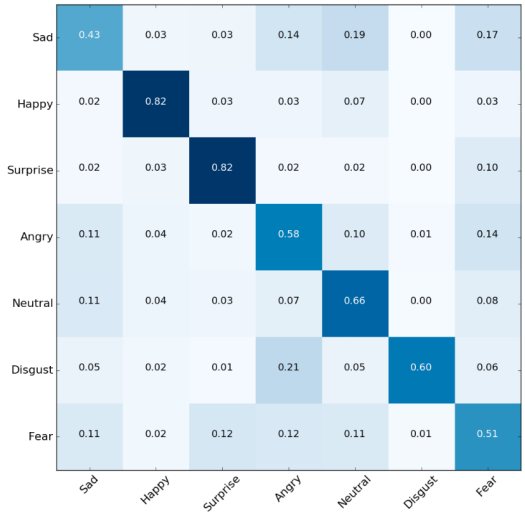


Fig. 3: Confusion matrix of the basic emotions in FER

V. User Interface

The target machine for the user interface is an Intel Core i3 4010 laptop accompanying the TurtleBot unit. It was not tested how well the CPU could handle forward passes of our CNN, but a magnitude of several seconds per pass can be safely assumed, as [16] reports 7.39 seconds for a forward pass on AlexNet with a significantly more powerful CPU. Due to the efficient implementation, the pre-processing step could be implemented directly on the laptop, while all CNN computations were outsourced to a server located separately. The server-side application opens a TCP port to receive pre-processed images and send the probabilities for each emotion. From the client's view, sending one image and receiving results took an average of 0.344 seconds when tested. Therefore, all data transfers happen asynchronously, ensuring responsiveness of the interface.

The server application uses the weights generated by training on all data, which subjectively gives the best results for a person facing the camera. The provocation of fear emotion was hard to detect which can be due to an insufficient amount of training data, the subjectivity of the supervising annotators or a sampling bias, since most of the training data were acquired by using the Google image search API without stratification. With an average frame rate of 26 per second on an Intel Core i5 3450 CPU, the performance can be considered satisfactory. Although it would have been desirable to run pre-processing and classification on a single machine, the client-server solution alleviates the need for expensive graphics cards in every unit running the software. The user interface can be seen in Fig. 4.

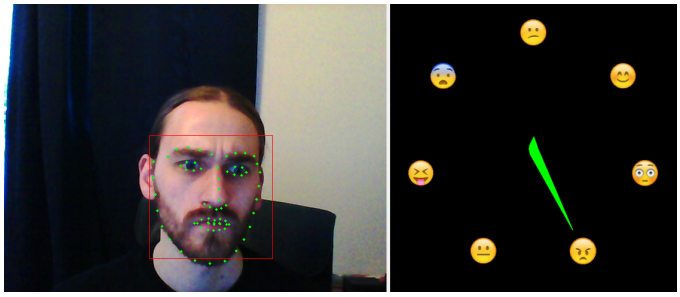


Fig. 4: The GUI showing anger emotion

VI. Conclusion and Perspectives

A state of the art algorithm capable of running on low-performance machines was composed. Further applications of the trained CNN lie in every software that requires emotion detection based on facial expressions, such as the development of learning applications that optimize the learner's flow [18] by detecting both anxiety and boredom or giving aid in teaching people with learning disabilities.

Pre-processing turned out to be a valuable step, enabling an admittedly deep, yet straightforwardly structured CNN to compete with models and algorithms using more complex approaches. Our pre-processing algorithm rectified faces by an affine transformation, but only a small amount of the available landmarks was used. A perspective transformation could yield a better result, although acquiring suitable points is more difficult. In addition, since the algorithm's use case is mainly video data, temporal information can also be included, e.g. by using a recurrent neural network. This is a great chance for future improvement regarding hard-to-separate emotions, like anger and neutrality or disgust and fear.

Acknowledgment

This work was supported by the Horizon 2020 funded project MaTHiSiS (Managing Affective-learning THrough Intelligent atoms and Smart InteractionS) nr. 687772 (http://www.mathisis-project.eu/). We gratefully acknowledge the support of NVIDIA Corporation with the donation of the Nvidia GeForce GTX TITAN X GPU used for this research.

References

[1] J. Schmidhuber, "Deep learning in neural networks: An overview," *Neural Networks*, vol. 61, pp. 85–117, 2015. 1

[2] O. M. Parkhi, A. Vedaldi, and A. Zisserman, "Deep face recognition," in *British Machine Vision Conference*, 2015. 1, 2

[3] Goodfellow *et al.*, "Challenges in representation learning: A report on three machine learning contests," in *International Conference on Neural Information Processing*. Springer, 2013, pp. 117–124. 1, 3

[4] Y. Tang, "Deep learning using support vector machines," *CoRR*, vol. abs/1306.0239, 2013. 1

[5] R. Bellman and R. Corporation, *Dynamic Programming*, ser. Rand Corporation research study. Princeton University Press, 1957. 1

[6] A. Dhall, R. Goecke, S. Lucey, and T. Gedeon, "Collecting large, richly annotated facial-expression databases from movies," *IEEE MultiMedia*, vol. 19, pp. 34–41, 2012. 1

[7] T. Kanade, J. F. Cohn, and Y. Tian, "Comprehensive database for facial expression analysis," in *Proceedings of the Fourth IEEE International Conference on Automatic Face and Gesture Recognition (FG'00)*, 2000, pp. 46–53. 1

[8] Lucey *et al.*, "The extended cohn-kanade dataset (ck+): A complete expression dataset for action unit and emotion-specified expression," in *Proceedings of the Third International Workshop on CVPR for Human Communicative Behavior Analysis (CVPR4HB 2010)*, 2010. 1

[9] D. Duncan, G. Shine, and C. English, "Facial emotion recognition in real time," Stanford University, Tech. Rep., 2016. 2

[10] G. Bradski and A. Kaehler, *Learning OpenCV: Computer vision with the OpenCV library*. " O'Reilly Media, Inc.", 2008. 2

[11] D. E. King, "Dlib-ml: A machine learning toolkit," *Journal of Machine Learning Research*, vol. 10, pp. 1755–1758, 2009. 2

[12] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," in *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, June 2005. 2

[13] D. E. King, "Max-margin object detection," *CoRR*, vol. abs/1502.00046, 2015. 2

[14] M. Danelljan, G. Hger, F. Shahbaz Khan, and M. Felsberg, "Accurate scale estimation for robust visual tracking," in *Proceedings of the British Machine Vision Conference*. BMVA Press, 2014. 2

[15] V. Kazemi and J. Sullivan, "One millisecond face alignment with an ensemble of regression trees," in *CVPR*, 2014. 2

[16] Jia *et al.*, "Caffe: Convolutional architecture for fast feature embedding," *arXiv preprint arXiv:1408.5093*, 2014. 3, 4

[17] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *CoRR*, vol. abs/1412.6980, 2014. 3

[18] S. J. Lopez, C. Snyder, J. Nakamura, and M. Csikszentmihalyi, "Flow theory and research," 2009. 4