Face Expression Recognition with a 2-Channel Convolutional Neural Network

Dennis Hamester, Pablo Barros, Stefan Wermter
University of Hamburg — Department of Informatics
Vogt-Kölln-Straße 30, 22527 Hamburg, Germany
http://www.informatik.uni-hamburg.de/WTM/
{hamester,barros,wermter}@informatik.uni-hamburg.de

***Abstract*—A new architecture based on the Multi-channel Convolutional Neural Network (MCCNN) is proposed for recognizing facial expressions. Two hard-coded feature extractors are replaced by a single channel which is partially trained in an unsupervised fashion as a Convolutional Autoencoder (CAE). One additional channel that contains a standard CNN is left unchanged. Information from both channels converges in a fully connected layer and is then used for classification. We perform two distinct experiments on the JAFFE dataset (leave-one-out and ten-fold cross validation) to evaluate our architecture. Our comparison with the previous model that uses hard-coded Sobel features shows that an additional channel of information with unsupervised learning can significantly boost accuracy and reduce the overall training time. Furthermore, experimental results are compared with benchmarks from the literature showing that our method provides state-of-the-art recognition rates for facial expressions. Our method outperforms previously published methods that used hand-crafted features by a large margin.**

I. Introduction

Recognizing human emotions has been the focus of attention in several areas from psychology and sociology to cognitive and computer science. Emotions have been studied for many years, and are still one of the most challenging topics in human psychological interactions. Through emotional state determination, it is possible to enhance, highlight and understand better human interactions, actions and even feelings.

The Artificial Intelligence community has studied and proposed approaches for automatic emotion recognition for the past two decades [1]–[3]. Emotional states can be conveyed by facial expressions, body posture, motion, language structure or even from the environment [4]. Based on facial expression, in one of the most comprehensive studies on understanding human emotions, Ekman and Friesen [5] established six universal emotions: *disgust*, *fear*, *happiness*, *surprise*, *sadness* and *anger*.

One of the most relevant characteristics of the presented systems is the use of a set of specific features, which constrains the solution to some rules. When used for facial expression recognition, the solutions based on hard-coded features show very good results, but lack the capability to be applied in a real-world scenario because of the many restrictions based on environment illumination, position of the subject, and skin color among others [6]. To overcome this problem in the computer vision area, the use of implicit features for image classification led to some success in the past years [7]–[9]. Among implicit feature models, deep neural networks [10] were proposed because they use the data themselves to learn the most significant aspects of the image. One of the advantages of deep neural networks is their power of generalization, and once they are trained, the low computational cost to extract features and classify them. Among deep neural architectures, deep belief networks [11] and Convolutional Neural Networks (CNN) [12] are the ones which present the most promising results for a wide range of applications [13]–[15].

In the past years, some approaches using deep neural networks for facial expression recognition were proposed and evaluated, for example by Kahou *et al.* [16]. In their work they evaluate two experiments using CNNs. In the first experiment, they implement a standard CNN and train it with the Acted Facial Expression in the Wild (AFEW) dataset [17]. In the second experiment, they pre-train the model with the Toronto Face Dataset. After that, a softmax layer is trained with the AFEW dataset. The networks presented in that approach implement the usual CNN model, adapting the number of layers and filters for each experiment executed. Both experiments were used for the recognition of six facial expressions, based on the universal facial expressions and one neutral expression. The idea of pre-training was shown to be successful, but a problem remains: the network is trained to learn specific features from the data it is shown. If a fair amount of data is shown to the network, it can have a significant capacity for generalization. If not, the network will not be able to learn proper features and will not be able to recognize images that were not shown during training.

Our proposed model uses a Multi-channel Convolutional Neural Network (MCCNN) architecture, proposed in previous work [18]. This network is able to learn with less data than the usual deep learning models, and needs less effort for training. One of the most sensitive restrictions of the MCCNN is the use of a pair of hard-coded Sobel-based layers which show good results when applied in a 3-channel topology. The proposed model replaces both Sobel channels with a single channel that is trained separately and unsupervised as a Convolutional Autoencoder (CAE).

Unsupervised feature extraction has become an attractive alternative to hand-crafted features and can yield highly competitive results [19], [20]. The second channel in our proposed

Hamester, D., Barros, P., Wermter, S. Face Expression Recognition with a 2-Channel Convolutional Neural Network. Proceedings of International Joint Conference on Neural Networks (IJCNN), pp. 1787-1794, Killarney, Ireland, July 2015. DOI: *missing*

model contains fixed filters that are learned by a CAE in a separate step. We train this layer on the Kyoto natural images dataset [21] so that the filters are not specific to our facial expression recognition task. These filter weights are kept fixed in all further task-specific training processes.

To evaluate the proposed model on a face recognition task, a set of experiments is described using the JAFFE dataset [22]. First, to find the most suitable model parameters, several parameter exploration experiments are designed, executed and evaluated. Using the selected parameters, two experiments with the JAFFE dataset are established. The results are collected, evaluated and discussed in this paper. Comparisons with benchmark results are also shown.

The paper is structured as follows: section II introduces our 2-channel architecture and describes preprocessing as well as how both channels are trained. The methodology for our experiments is given in section III. Preliminary experiments for parameter optimization and the results are shown as well. Section IV shows results from our main experiments and compares them to related work. Finally, a conclusion is given in section V.

II. Architecture

The MCCNN architecture uses multiple independent Convolutional Neural Networks in parallel, but connects them in the last layer to extract and recognize patterns from images. In previous approaches, the use of three channels was explored, and presented good results for hand posture [18] and motion recognition [23]. The concept behind this architecture is to make use of existing knowledge, here represented by a different channel, to diversify the input. In our previous work, each channel received different information, usually Sobel-based filters were applied to the image. The channels receiving the image after the application of the Sobel operators acted as tuning for the layer receiving the raw image. At the end, all channels were connected and able to extract different and specific information of the image.

The Sobel-based layers were a solution to explore the edges in two directions, horizontal and vertical. To extend this principle, our new proposed model (see Figure 1) replaces the hard-coded Sobel-based layers by a layer containing Gabor-like filters. This increases the kind of information extracted from the image, specializing one channel with a strong multiple-orientation edge detector. Following the multi-channel architecture, the proposed model uses two channels, both of them receiving the same image. What differs in the channels is the implementation and training of the filters in the first layer of each channel. The first channel acts like a common CNN training all the parameters in a supervised fashion. The second channel uses pre-trained parameters, obtained by a CAE, which learn Gabor-like filters. After pre-training, the weights of this layer are fixed and not trained in the supervised phase. The two channels are connected with a fully-connected hidden layer that produces the output for a logistic regression classifier. All parameters of network are optimized at the same time. The final error is obtained with the output of the two channels, so that both second layers act like a bias for each other. Thus, the final classification layer in our proposed model combines information from both channels, a standard CNN and a second channel whose weights are trained as CAE. Figure 1 shows the architecture of the model which was used in the experiments.

When training a CAE on natural images, the resulting filters usually look like Gabor filters, which respond strongly to local edge elements. They are thus similar to hard-coded Sobel filters used in our previous work. However, the Gabor-like filters here are much more diverse in the sense that there are filters for many more specific orientations. Subsequent layers in our architecture receive explicit information about the existence of specific edges making classification later on easier and more robust.

This approach provides an important advantage over pure supervised methods besides a potential boost in recognition performance. After training a CAE once on a generic dataset such as the Kyoto database [21], the network can be transferred to several different tasks. Reusing a pre-trained network for the first layer in a potentially deep architecture is a viable method to speed up overall training time.

A. Channel 1 — Convolutional Neural Network

A Convolutional Neural Network is a set of pairs of convolution and max-pooling layers that enable the model to extract and enhance implicit features of an image. When stacked together, the first layers act like an edge enhancement and allow to extract local features, which are passed to deeper layers that act as complex feature extractors.

Each convolutional layer contains a set of feature maps, or filters, that extract features from a region of units using a convolution operation. Then, an additive bias is applied and the result is passed through a non-linear activation function. The value of a unit v_{nc}^{xy} in the position (x,y) at the n-th feature map in the c-th layer is given by

$$v_{nc}^{xy} = \max\left(b_{cn} + \sum_{m}\sum_{h=0}^{H_{i-1}}\sum_{w=0}^{W_{i-1}} w_{ijm}^{hw} v_{(i-1)m}^{(x+h)(y+w)}, 0\right) \quad (1)$$

where $\max(\cdot, 0)$ represents the rectified linear function, which was shown to be more suitable for training deep neural architectures [24], b_{cn} is the bias for the n-th feature map of the c-th layer, and m indexes over the set of features maps in the (i-1) layer connected to the current layer c. In the equation, w_{nck} is the weight of the connection between the unit (h,w) within a region, or kernel, connected to the previous layer. H_i and W_i are the height and width of the kernel.

In the max-pooling layers, a region of the previous layer is connected to a unit in the current layer, reducing the dimensionality of the feature maps. Each max-pooling layer retains only the maximum value within its receptive field and passes it to the next layer. This enhances invariance to scale and distortions of the input [9]. The parameters of a CNN could be learned either by a supervised approach tuning the filters in

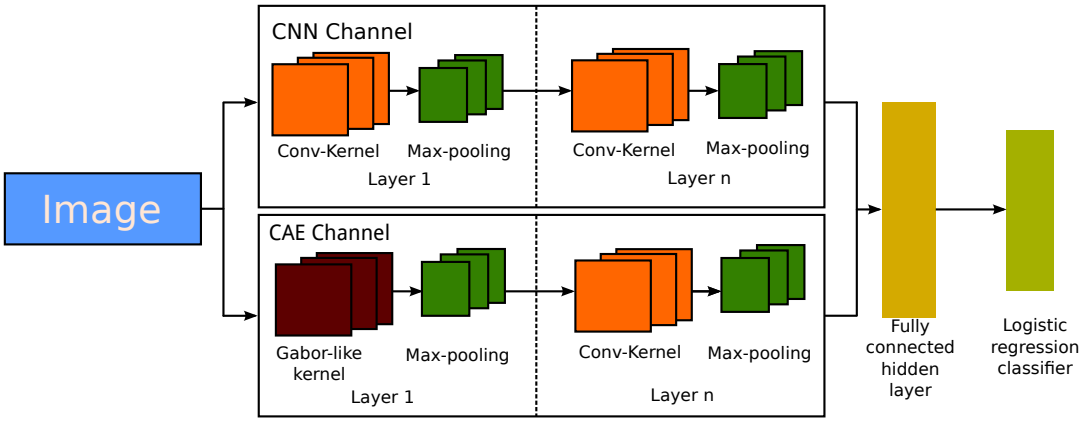


Figure 1. Proposed architecture for a Multi-channel Convolutional Neural Network using 2 channels. The upper channel is a standard CNN, consisting of convolutional layers and max-pooling layers. The lower channel has the same topology, but its first layer weights are trained as a Convolutional Autoencoder.

a training database [11], or by an unsupervised approach [19]. The proposed model uses the supervised approach.

B. Channel 2 — Convolutional Autoencoder

The second channel in our proposed architecture consists of two convolutional layers with max pooling in between. The input to this channel is the same as for the first channel. However, the first layer is trained as a Convolutional Autoencoder in an extra step, prior to supervised training of the whole network. This step is similar to pre-training found in other deep-learning methods [11]. However, we keep the weights of the first layer fixed during the supervised learning phase, whereas pre-training is usually followed by supervised fine-tuning.

The hidden layer uses rectified linear units as well and activations in the i-th feature map are given by:

$$\mathbf{y}_i = \max\left(\mathbf{x} * W_i + b_i, 0\right) \tag{2}$$

Reconstructions are computed as linear combinations of $\mathbf{y}_i$ and use tied weights ($\widetilde{W}$ denotes the vertically and horizontally flipped version of W):

$$\hat{\mathbf{x}} = \sum_i \left[\mathbf{y}_i * \widetilde{W}_i\right] + b' \tag{3}$$

A CAE with k feature maps is over-complete by a factor of roughly k[1]. In order to prevent the network from learning trivial solutions like identity functions, the capacity of the hidden layer must be regularized during training. We impose a very strong sparsity prior by regularizing the network using a Winner-Take-All (WTA) method, as described by Makhzani and Frey [25]. For each feature map, the maximum value is determined after applying the rectified linear function. This value is retained, but all other values in the feature map are set to zero. This can be thought of as multiplying each feature map with a different mask that leaves only one value intact before computing the reconstructions.

[1]The exact size of a feature maps depends on the border treatment of the convolution.

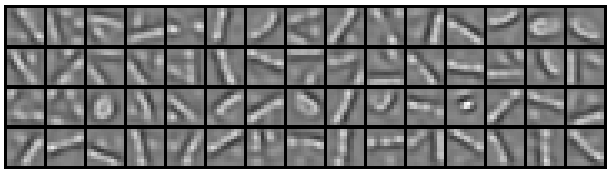


Figure 2. Typical filters learned by our Convolutional Autoencoder with Winner-Take-All regularization.

This approach is very efficient, especially in comparison with methods like contractive regularization [26]. Since the WTA approach avoids computing additional gradients (or Jacobian matrices) and relies only on determining the maximum in each feature map. We also found this method to be highly effective at learning sparse codes while not introducing more hyper-parameters like trade-off weights in the case of contractive regularization or many other regularization techniques. WTA regularization is only applied during training and dropped after the network is finalized and the weights are frozen. Otherwise, the hidden layer would contain only a single non-zero value per feature map and be incapable of properly transferring information about the whole image towards the next layer.

Makhzani and Frey [25] state that using the same weights for the hidden layer as well as for the reconstruction (tied weights) hurts generalization. However, we use tied weights as shown in eqs. (2) and (3) and observe no such problems in any of our experiments. On the other hand, having two distinct weight matrices (or tensors in the case of convolutional networks) doubles the number of model parameters and makes training more difficult and more time consuming, which is why we decided to use tied weights in the first place.

The CAE is trained on the Kyoto natural images dataset [21] using a fixed kernel of 11×11 pixels. Each image is pre-processed by convolving it with a Difference of Gaussian (DoG) filter. DoG filters are effectively similar to ZCA whitening [27], without the need to learn the filter kernels first. Furthermore, the shape of DoG filters is a good approximation

Figure 3. Samples of all seven classes from the JAFFE dataset [22]. The two-letter codes in parentheses are used throughout this paper to refer to the class. Each sample shows a different subject as well, out of the ten subjects in the dataset.

of ideal decorrelation filters for grayscale images [28]. Whitening enables our CAE to learn Gabor-like filters, otherwise filters would look more like principle components [29]. In comparison to ZCA whitening, using DoG filters in the context of convolutional networks also has the advantage that they can be applied directly to arbitrarily-sized images. The number of features is not set a-priori but determined experimentally. Details are given in section III. Figure 2 shows 60 typical Gabor-like filters learned by our CAE.

III. Experiments

We evaluate our architecture for facial expression recognition on the well established JAFFE dataset [22]. The dataset consists of 213 images, showing ten Japanese women performing seven different facial expressions (*neutral*, *happiness*, *sadness*, *surprise*, *anger*, *disgust*, and *fear*). For each combination of subject and facial expression, there are between two and four examples present in the dataset. All images have an equal resolution of 256×256 pixels and are in grayscale. We usually refer to the seven classes using common two-letter codes (i.e. NE for *neutral*) as shown in fig. 3.

Even though the JAFFE dataset was first published in 1998 [22], it is still relevant today and used as a benchmark for facial expression recognition throughout the literature [30]–[32]. It is a challenging dataset because there are only few examples per subject / expression, even more so if a separate testing set is excluded from training. Furthermore, the fact that every subject performs every expression allows for interesting experiments like for example the leave-one-out scheme described later in this section.

The performance is analyzed with two different experiments following the methodology of Subramanian *et al.* [32]. In the first experiment, a leave-one-out scheme is used to split the dataset into training and testing subsets. One sample from each subject / expression combination is randomly taken out and used for testing. In total, there are 143 training samples and 70 samples for testing [32]. This process is repeated 15 times and performance measures are averaged.

In the second experiment, we use ten-fold cross validation to determine the training and testing sets. The whole dataset is randomly split into ten almost equally sized subsets of about 21 samples each. Each subset is then used once for testing while the remaining nine subsets compose the training data. Finally, the performance measures of all ten splits are averaged. As with the first experiment, this process is repeated 15 times.

Table I
Classification F_1-scores, standard deviations and training time for five different numbers of feature maps in the CAE layer. The time measures a complete run of ten-fold cross validation.

Feature maps	20	40	60	80	100
Mean F_1-score	89.8%	91.2%	91.8%	93.7%	90.4%
Std. dev.	4.3%	9%	6.3%	2.6%	3.9%
Time (Minutes)	16.4	25	41.1	58.4	85.4

As mentioned in section II, we optimize the number of feature maps in the CAE and both first layers in the 3-channel Sobel-based network. In both cases, we explore a range from 20 to 100 feature maps, in steps of 20. For this parameter exploration we use the same ten-fold cross validation as it is used in the second main experiment. For each parameter, the average F_1-score over all classes is determined, of which in turn the mean is computed after running all ten partitions in the cross validation.

Table I shows average F_1-scores for the explored range as well as standard deviations. The general trend is an increase in performance as more feature maps are used, which is not surprising. However, the F_1-score drops to 90.4% on 100 feature maps, which might indicate that the large amount of model parameters causes training to be more difficult. Even as low as 20 feature maps result in a classifier that achieves just short of 90% F_1-score. This is a very good result, considering that training a corresponding network takes less than two minutes[2].

For our Sobel-based network, the results are shown in table II. Interestingly, a network that contains only 20 feature maps in its first layers performs better than all other parameters we tested. It is also noteworthy, that this particular network performs better than our equivalent CAE-based network (compare table I), which is not true for any of the other parameter values. In general, it seems the Sobel-based architecture has difficulties during training when too many feature maps and consequently too many model parameters are present.

In order to determine the network parameters for our final experiments, we are not just taking into account the F_1-scores of the presented ten-fold cross validation, but also the time it takes to train a network. The goal is to find a suitable compromise between both criteria. Considering the training time is important here, because both main experiments are repeated 15 times for each network architecture (CAE- and Sobel-based). Tables I and II show that the training time

[2]This value corresponds to training a single network, not ten-fold cross validation, and excludes the time it took to train the CAE.

Table II
CLASSIFICATION F_1-SCORES, STANDARD DEVIATIONS AND TRAINING TIME FOR FIVE DIFFERENT NUMBERS OF FEATURE MAPS IN BOTH SOBEL LAYERS. THE TIME MEASURES A COMPLETE RUN OF TEN-FOLD CROSS VALIDATION.

Feature maps	20	40	60	80	100
Mean F_1-score	93.4%	89.5%	88%	90.8%	88%
Std. dev.	2%	5.8%	6%	5.8%	5.7%
Time (Minutes)	21.5	34.6	60.1	88.1	130.1

depends roughly linear on the number of features and ranges from as low as 16.4 minutes to more than 2 hours. Based on these considerations as well as the F_1-scores discussed before, we choose 40 feature maps for both architectures and all following experiments. Choosing a value on the lower half of the range allows us repeat our main experiments often, because they can be trained quickly. On the other hand, both models achieve good F_1-scores (about 90% and 91%), which are well within the average of the considered range here. Choosing the same number of feature maps for both models also has the advantage of better comparability.

Of course, our method requires determining several hyper-parameters. These include the number of layers in each channel, their number of feature maps and kernel sizes, as well as the size of the fully-connected layer that merges both channels. We concentrate our efforts here on the layer that is trained in an unsupervised fashion, mainly because the MCCNN architecture itself has already been explored in previous work [18]. The same parameter is optimized in our Sobel-based network in order to provide an equal ground for all following experiments.

All experiments as well as the CAE training were performed on a desktop machine with an Intel Xeon E5630 processor, two Nvidia GeForce GTX 590 graphics cards and 24 GiB of RAM. Theano was used to accelerate computations on both GPUs [33], [34]. Training the CAE with 40 feature maps took about 15 minutes and comprised 20000 parameter updates. Training times for the whole network are given in Tables I and II.

IV. RESULTS

A. Experiment 1

We first report the results of the leave-one-out experiment, averaged over 15 runs. Table III shows the summary of our results, as well as the results reported by Subramanian *et al.* [32]. For our experiment, an average accuracy of 95.8% was obtained with a standard deviation of 1.6. These results present an improvement of almost 8% when compared to the McFIS method. When compared with the architecture using Sobel-based filters, the results are almost 3% higher. While the CAE-based network consists of 2 channels, the Sobel-based one is implemented with 3 channels. The training and recognition time are improved as less parameters need to be updated, while at the same time, more diverse information is present.

Table III
RESULTS AND COMPARISON FOR THE LEAVE-ONE-OUT EXPERIMENT. REPORTED ARE AVERAGE ACCURACIES AND STANDARD DEVIATIONS.

Method	Performance
CAE-based	95.8 ± 1.6
Sobel-based	93.1 ± 1.6
McFIS [32]	87.6 ± 5.79

Table IV
COMBINED CONFUSION MATRIX OF 15 LEAVE-ONE-OUT EXPERIMENTS. VALUES ARE GIVEN IN PERCENT AND DO NOT NECESSARILY ADD UP TO 100% DUE TO ROUNDING.

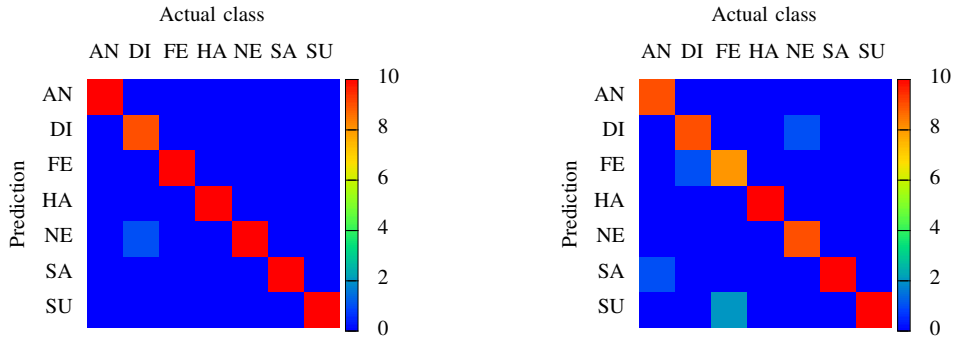
		Actual class						
		AN	DI	FE	HA	NE	SA	SU
Prediction	AN	94	0	2.7	0	0.7	0	0
	DI	0	97.3	2.7	0	4.7	0	0
	FE	0	0.7	90.7	0	0	0	0
	HA	0	0	0	99.3	1.3	0	0
	NE	0	2	0	0	93.3	0	4
	SA	6	0	0	0	0	100	0
	SU	0	0	4	0.7	0	0	96

A combined confusion matrix is shown in table IV. Each cell in the matrix is computed as the average of the corresponding values of all confusion matrices. The low standard deviations are visible here as well, as most values are either zero or quite low. Especially noticeable are the results for the class *fear* (FE). The JAFFE dataset authors describe fear as being difficult to express for Japanese people and consequently exclude it from several experiments. This is consistent with our results here, as *fear* has the lowest accuracy (90.7%) among all classes.

In fig. 4 we show the best and worst confusion matrices based on the accuracy. In the best case we achieved an accuracy of 98.6%. In this particular experiment, only a single sample of the training set was misclassified (*neutral* instead of *disgust*). Subramanian *et al.* [32] report a best result of 94.5% accuracy which is about 4% lower than ours. The right image in fig. 4 shows the worst single experiment in which we still achieved 92.9% accuracy. Here we can see that three classes (*happiness*, *sadness* and *surprise*) were classified perfectly. All other classes have only minor misclassification (at most 2 out of 10).

B. Experiment 2

The second experiment uses ten-fold cross validation and is repeated for 15 runs as well. A summary of the results is shown in table V. The same table also shows results from the literature for comparison. We achieve an accuracy of 94.1% with a standard deviation of 4.3. Our CAE-based MCCNN architecture achieves more than 5% better accuracy than the McFIS method [32] and is on average slightly better the Linear Programming method presented by Feng *et al.* [35].



(a) Best confusion matrix

(b) Worst confusion matrix

Figure 4. Best (a) and worst (b) confusion matrices based on accuracies of a single leave-one-out experiment.

Table V
RESULTS AND COMPARISON FOR THE TEN-FOLD CROSS VALIDATION EXPERIMENT. REPORTED ARE AVERAGE ACCURACIES AND STANDARD DEVIATIONS.

Method	Performance
CAE-based	94.1 ± 4.3
Sobel-based	92 ± 6.1
McFIS [32]	89.05 ± 3.214
Linear Programming [35]	93.8

The improvement over the Sobel-based network is not as high (2.1%) but still consistent. It should be noted, that the CAE-based network has one channel less than the Sobel-based network, whereas the remaining topology is identical. It might thus be concluded that the proposed method is less expressive. However as the experiments show, the Gabor-like filters in our network support classification very well.

Interestingly, the accuracy here is lower than for the leave-one-out experiment reported earlier. At the same time, the standard deviation is much higher (4.3% vs. 1.6%). We believe this is a result of ambiguities in the dataset [22] and the small (10%) testing set used here, compared to the previous experiment (about 33%). There is a higher chance that a certain class is represented by only ambiguous samples, which might account for the higher standard deviation as well as a slightly lower average accuracy.

In table VI, we show a combined confusion matrix of all ten-fold cross validation experiments. Compared to the first experiment (table IV), the higher standard deviation is visible as many more cells are now greater than zero. However, many cells outside the main diagonal are just slightly above zero which means that most of the time very few sample were misclassified. The results are consistent with the previous ones regarding *fear*, which shows the lowest accuracy again (85.9%). Judging by both experiments (tables IV and VI), *fear* is mostly confused with *anger*, *disgust* and *surprise*.

Table VI
COMBINED CONFUSION MATRIX OF THE TEN-FOLD CROSS VALIDATION EXPERIMENTS. VALUES ARE GIVEN IN PERCENT AND DO NOT NECESSARILY ADD UP TO 100% DUE TO ROUNDING.

		Actual class						
		AN	DI	FE	HA	NE	SA	SU
Prediction	AN	90.9	0.8	3	0	1.4	2.5	0
	DI	0	89.6	4.3	0.8	4.9	0	0
	FE	1.1	4.8	85.9	0	0	0	1.7
	HA	0	2.4	0.5	96.3	3.7	0	0.9
	NE	0	2.1	0.8	0	88	0	4.3
	SA	8.1	0.3	0.5	0.5	0	97.5	0
	SU	0	0	5	2.4	2	0	93.1

V. CONCLUSION

In this paper, we presented a variant of the Multi-Channel Convolutional Neural Network. The architecture is characterized by multiple CNNs (channels), that first process information independently. All streams merge in a fully-connected layer, which is used for classification. Compared to previous work [18], hard-coded Sobel filters are replaced by a single channel, making the proposed model a 2-channel architecture. The weights of this new channel are trained as Convolutional Autoencoder, which usually results in Gabor-like filters when trained on whitened natural images. This model is motivated by two observations: First, using additional channels with input from edge detectors (Sobel) proved useful in previous work. Our Gabor-like filters provide qualitatively the same information, but in a much more diverse scope, because the CAE has 40 different feature maps. Second, the effort it takes to fully train a CAE-based network is lower than the previous Sobel-based network. Having one channel less certainly contributes to this, but also the fact that the CAE layer is reusable. The time it takes to train a CAE can easily be amortized by using it multiple times.

Evaluation of our model was done using the JAFFE dataset

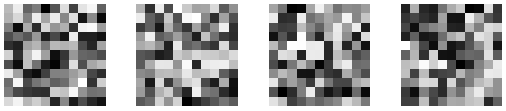


Figure 5. Visualizations of four first-layer filters of the CNN channel.

and we followed the methodology of two experiments done by Subramanian *et al.* [32]. Both experiments were performed on the proposed model as well as the Sobel-based model for comparison. The experiments and results show, that the proposed model is a very viable method for facial expression recognition. It does not depend on any hand-crafted or task-specific feature extraction but exploits unsupervised learning. We easily reach state-of-the-art recognition rates with minimal effort. In the leave-one-out experiment, we achieve an average of 95.8% accuracy, which is 8.2% higher than what Subramanian *et al.* [32] reported. The ten-fold cross validation leads to an average accuracy of 94.1%.

The improvements over our previous Sobel-based model range between two and three percent. Thus, our new model does not only perform better in terms of classification performance but is also faster to train, because it uses one channel less. However, the first layer in the CAE channel has to be taken into account as well, when considering training time, because it is trained in a separate step and the CAE training effort pays off the more often it is reused.

For future work, we would like to investigate what exact role each channel plays during the implicit feature extraction task. The CAE channel for example already contains diverse information about the existence of edges. It seems intuitive to assume, that the first channel (which is trained fully supervised) would not take on the same filter shapes. In this sense, it is still unclear in what way both channels influence each other and whether they extract complementary information. Another possibility might be that both channels learn qualitatively the same information and act as weak classifiers. In this case, the fully-connected layer at the end might act as a boosting layer. In some very preliminary experiments to understand better what our architecture learns, we visualized the first layer filters of the channel, which is trained fully supervised (see fig. 5). On first sight and especially in comparison to the CAE filters (compare Figure 2), the filter do not seem to have meaningful structure after learning. On the other hand, the filter do not just contain white noise. A possibly better approach, that we want to pursue in the future, might be to visualize not only first layer filters but also those in other layers. For example, the deconvolutional networks used by Zeiler and Fergus [36] could give valuable insights into our networks. However, even without having analyzed the relationship between both channels yet, our model achieves state-of-the-art recognition rates for face expressions and is very fast to train.

ACKNOWLEDGEMENTS

This work was partially supported by the State Graduate Funding Program at the University of Hamburg and by CAPES Brazilian Federal Agency for the Support and Evaluation of Graduate Education (p.n.5951–13–5).

The authors thank Dr. Cornelius Weber and Katja Kösters for their valuable suggestions when writing this paper.

REFERENCES

[1] S. Morishima and H. Harashima, "Facial expression synthesis based on natural voice for virtual face-to-face communication with machine", in *, 1993 IEEE Virtual Reality Annual International Symposium, 1993*, Sep. 1993, pp. 486–491.

[2] A. Colmenarez, B. Frey, and T. Huang, "A probabilistic framework for embedded face and facial expression recognition", in *Computer Vision and Pattern Recognition, 1999. IEEE Computer Society Conference on.*, vol. 1, 1999.

[3] B. Chu, S. Romdhani, and L. Chen, "3D-Aided Face Recognition Robust to Expression and Pose Variations", in *2014 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun. 2014, pp. 1907–1914.

[4] M. Cabanac, "What is emotion?", *Behavioural Processes*, vol. 60, no. 2, pp. 69–83, Nov. 2002.

[5] P. Ekman and W. V. Friesen, "Constants across cultures in the face and emotion", *J. Pers. Soc. Psychol.*, vol. 17, no. 2, pp. 124–129, 1971.

[6] Z. Guo-Feng, W. Ke-Jun, Y. Lei, and F. Gui-Xia, "New research advances in facial expression recognition", in *Control and Decision Conference (CCDC), 2013 25th Chinese*, May 2013, pp. 3403–3409.

[7] J. Nagi, F. Ducatelle, G. Di Caro, D. Ciresan, U. Meier, A. Giusti, F. Nagi, J. Schmidhuber, and L. Gambardella, "Max-pooling convolutional neural networks for vision-based hand gesture recognition", in *2011 IEEE International Conference on Signal and Image Processing Applications (ICSIPA)*, Nov. 2011, pp. 342–347.

[8] S. Ji, W. Xu, M. Yang, and K. Yu, "3D Convolutional Neural Networks for Human Action Recognition", *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 35, no. 1, pp. 221–231, Jan. 2013.

[9] D. Ciresan, U. Meier, and J. Schmidhuber, "Multi-column deep neural networks for image classification", in *2012 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun. 2012, pp. 3642–3649.

[10] Y. Bengio, A. Courville, and P. Vincent, "Representation Learning: A Review and New Perspectives", *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 35, no. 8, pp. 1798–1828, Aug. 2013.

[11] G. Hinton, S. Osindero, and Y. Teh, "A Fast Learning Algorithm for Deep Belief Nets", *Neural Comput.*, vol. 18, no. 7, pp. 1527–1554, Jul. 2006.

[12] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition", *Proc. IEEE*, vol. 86, no. 11, pp. 2278–2324, Nov. 1998.

[13] S. Lawrence, C. Giles, A. C. Tsoi, and A. Back, "Face recognition: a convolutional neural-network approach", *IEEE Trans. Neural Netw.*, vol. 8, no. 1, pp. 98–113, Jan. 1997.

[14] M. Khalil-Hani and L. S. Sung, "A convolutional neural network approach for face verification", in *2014 International Conference on High Performance Computing Simulation (HPCS)*, Jul. 2014, pp. 707–714.

[15] T. P. Karnowski, I. Arel, and D. Rose, "Deep Spatiotemporal Feature Learning with Application to Image Classification", in *2010 Ninth International Conference on Machine Learning and Applications (ICMLA)*, Dec. 2010, pp. 883–888.

[16] S. E. Kahou, C. Pal, X. Bouthillier, P. Froumenty, Ç. Gülçehre, R. Memisevic, P. Vincent, A. Courville, Y. Bengio, R. C. Ferrari, M. Mirza, S. Jean, P.-L. Carrier, Y. Dauphin, N. Boulanger-Lewandowski, A. Aggarwal, J. Zumer, P. Lamblin, J.-P. Raymond, G. Desjardins, R. Pascanu, D. Warde-Farley, A. Torabi, A. Sharma, E. Bengio, M. Côté, K. R. Konda, and Z. Wu, "Combining Modality Specific Deep Neural Networks for Emotion Recognition in Video", in *Proceedings of the 15th ACM on International Conference on Multimodal Interaction*, ser. ICMI '13, New York, NY, USA: ACM, 2013, pp. 543–550.

[17] A. Dhall, R. Goecke, S. Lucey, and T. Gedeon, "Collecting Large, Richly Annotated Facial-Expression Databases from Movies", *IEEE Multimedia*, vol. 19, no. 3, pp. 34–41, 2012.

[18] P. Barros, S. Magg, C. Weber, and S. Wermter, "A Multichannel Convolutional Neural Network for Hand Posture Recognition", in *Artificial Neural Networks and Machine Learning – ICANN 2014*, S. Wermter, C. Weber, W. Duch, T. Honkela, P. Koprinkova-Hristova, S. Magg, G. Palm, and A. E. P. Villa, Eds., ser. Lecture Notes in Computer Science 8681. Springer International Publishing, Sep. 15, 2014, pp. 403–410.

[19] M. Ranzato, F. J. Huang, Y.-L. Boureau, and Y. LeCun, "Unsupervised Learning of Invariant Feature Hierarchies with Applications to Object Recognition", in *IEEE Conference on Computer Vision and Pattern Recognition, 2007. CVPR '07*, Jun. 2007, pp. 1–8.

[20] M. Ranzato, R. Monga, M. Devin, K. Chen, G. Corrado, J. Dean, Q. V. Le, and A. Y. Ng, "Building high-level features using large scale unsupervised learning", presented at the Proceedings of the 29th International Conference on Machine Learning (ICML-12), 2012, pp. 81–88.

[21] E. Doi, T. Inui, T.-W. Lee, T. Wachtler, and T. J. Sejnowski, "Spatiochromatic Receptive Field Properties Derived from Information-Theoretic Analyses of Cone Mosaic Responses to Natural Scenes", *Neural Computation*, vol. 15, no. 2, pp. 397–417, Feb. 1, 2003.

[22] M. Lyons, S. Akamatsu, M. Kamachi, and J. Gyoba, "Coding facial expressions with Gabor wavelets", in *Third IEEE International Conference on Automatic Face and Gesture Recognition, 1998. Proceedings*, Apr. 1998, pp. 200–205.

[23] P. Barros, G. I. Parisi, D. Jirak, and S. Wermter, "Real-time Gesture Recognition Using a Humanoid Robot with a Deep Neural Architecture", in *Humanoid Robots (Humanoids), 2014 14th IEEE-RAS International Conference on*, Nov. 2014, In Press.

[24] X. Glorot, A. Bordes, and Y. Bengio, "Deep Sparse Rectifier Neural Networks", presented at the International Conference on Artificial Intelligence and Statistics, 2011, pp. 315–323.

[25] A. Makhzani and B. Frey, "A Winner-Take-All Method for Training Sparse Convolutional Autoencoders", *ArXiv14092752 Cs*, Sep. 9, 2014, arXiv: 1409.2752.

[26] S. Rifai, P. Vincent, X. Muller, X. Glorot, and Y. Bengio, "Contractive Auto-Encoders: Explicit Invariance During Feature Extraction", presented at the Proceedings of the 28th International Conference on Machine Learning (ICML-11), 2011, pp. 833–840.

[27] A. J. Bell and T. J. Sejnowski, "The "independent components" of natural scenes are edge filters", *Vision Research*, vol. 37, no. 23, pp. 3327–3338, Dec. 1997.

[28] M. Brown, S. Susstrunk, and P. Fua, "Spatio-chromatic decorrelation by shift-invariant filtering", in *2011 IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, Jun. 2011, pp. 27–34.

[29] A. Coates, A. Y. Ng, and H. Lee, "An Analysis of Single-Layer Networks in Unsupervised Feature Learning", presented at the International Conference on Artificial Intelligence and Statistics, 2011, pp. 215–223.

[30] F. Cheng, J. Yu, and H. Xiong, "Facial Expression Recognition in JAFFE Dataset Based on Gaussian Process Classification", *IEEE Trans. Neural Netw.*, vol. 21, no. 10, pp. 1685–1690, Oct. 2010.

[31] S. Liu, Y. Zhang, and K. Liu, "Facial expression recognition under random block occlusion based on maximum likelihood estimation sparse representation", in *2014 International Joint Conference on Neural Networks (IJCNN)*, Jul. 2014, pp. 1285–1290.

[32] K. Subramanian, S. Suresh, and R. Venkatesh Babu, "Meta-Cognitive Neuro-Fuzzy Inference System for human emotion recognition", in *The 2012 International Joint Conference on Neural Networks (IJCNN)*, Jun. 2012, pp. 1–7.

[33] J. Bergstra, O. Breuleux, F. Bastien, P. Lamblin, R. Pascanu, G. Desjardins, J. Turian, D. Warde-Farley, and Y. Bengio, "Theano: A CPU and GPU Math Compiler in Python", presented at the Proceedings of the 9th Python in Science Conference, 2010, pp. 3–10.

[34] F. Bastien, P. Lamblin, R. Pascanu, J. Bergstra, I. Goodfellow, A. Bergeron, N. Bouchard, D. Warde-Farley, and Y. Bengio, "Theano: new features and speed improvements", *ArXiv12115590 Cs*, Nov. 23, 2012, arXiv: 1211.5590.

[35] X. Feng, M. Pietikäinen, and A. Hadid, "Facial expression recognition with local binary patterns and linear programming", *Pattern Recognit. Image Anal.*, vol. 15, no. 2, pp. 546–548, 2005.

[36] M. D. Zeiler and R. Fergus, "Visualizing and Understanding Convolutional Networks", in *Computer Vision – ECCV 2014*, D. Fleet, T. Pajdla, B. Schiele, and T. Tuytelaars, Eds., ser. Lecture Notes in Computer Science 8689. Springer International Publishing, Jan. 1, 2014, pp. 818–833.