In: Proc. of the IEEE International Joint Conference on Neural Networks (IJCNN/WCCI), Vancouver, Canada (July, 2016)

Understanding How Deep Neural Networks Learn Face Expressions

Nima Mousavi
Department of Computer Science
University of Hamburg
Hamburg, Germany
8mousavi@informatik.uni-hamburg.de

Henrique Siqueira
Polytechnic School
University of Pernambuco
Recife, Brazil
hcs@ecomp.poli.br

Pablo Barros
Department of Computer Science
University of Hamburg
Hamburg, Germany
barros@informatik.uni-hamburg.de

Bruno Fernandes
Polytechnic School
University of Pernambuco
Recife, Brazil
bjtf@ecomp.poli.br

Stefan Wermter
Department of Computer Science
University of Hamburg
Hamburg, Germany
wermter@informatik.uni-hamburg.de

***Abstract*—Deep neural networks have been used successfully for several different computer vision-related tasks, including facial expression recognition. In spite of the good results, it is still not clear why these networks achieve such good recognition rates. One way to learn more about deep neural networks is to visualise and understand what they are learning, and to do so techniques such as deconvolution could play a significant role. In this paper, we train a Convolutional Neural Network (CNN) and Lateral Inhibition Pyramidal Neural Network (LIPNet) to learn facial expressions. Then, we use the deconvolution process to visualise the learned features of the CNN and we introduce a novel mechanism for visualising the internal representation of the LIPNet. We perform a series of experiments, training our networks with the Cohn-Kanade data set and show what kind of facial structures compose the learned emotion expression representation. Then, we use the trained networks to recognise images from the Jaffe data set and demonstrate that the learned representations are present in different face images, emphasizing the generalization aspects of these networks. We discuss the different representations that each network learns and how they differ from each other. We also discuss how each learned representation contributes to the recognition process and how they can be compared to the emotional notation Facial Action Coding System - Facs. Finally, we explain how the principles of invariance, redundancy and filtering, common for deep networks, contribute to the learned features and to the facial expression recognition task in general.**

I. Introduction

The analysis of human emotions through physical expressions is deeply rooted in psychological research [1]. In the course of studying emotions and their connection to facial expressions, Ekman [2] developed the **F**acial **A**ction **C**oding **S**ystem (FACS). FACS is a method, developed from human observers, to categorize facial expressions in terms of emotions. Expressions are categorized into action units (actions of individual muscles) and rated in terms of intensity.

Likewise, this means that automated systems that detect facial expressions could be used to make hypothesis about the emotions of a subject. Especially long-time monitoring of subjects in extreme situations (e.g. space exploration [3]) might be examined through the aid of computers [4], due to the circumstances of not being in direct contact with the subjects. Valstar et al. [5] propose such an automated system, which is related to the FACS-model to classify face expressions. They propose to detect action units, from which facial expressions can be inferred through FACS.

Problems with the FACS-based approach include a complex implementation and a possible long runtime, since the architecture needs multiple stages to detect multiple action units. Another problem involves the datasets, as it might be difficult to get an adequate database annotated in terms of action units. However, the biggest problem is caused by the restriction of the FACS-model itself, since it affects its generalization capability. The analysed action units are restricted to the movement of facial muscles and an automated system based on FACS has to infer this movement based on visual features. These features must be compared with a code of action units which will determine the face expression. This process restricts the execution of each expression, not allowing the space for different persons performing the same expression in different ways, which happens in natural scenarios. For example, the same person can have different ways of smiling to express its happiness, or even smile to express sarcasm.

Deep neural networks that work on data representations, such as **C**onvolutional **N**eural **N**etworks (CNN) used in [6], have a conceptual similarity to the one used to build the FACS model. The input is decomposed into features and each deeper layer has a more complex representation, which builds upon the previous layer. The final feature representations are then used for classification.

But in contrast to architectures built around FACS, these neural networks are not constrained in any way. They learn with the data that is presented to the network and adapt to it.

This means that as soon as the network sees different persons smiling, the network can build a complex feature that will be able to generalize the smile action, and then categorize it into an emotional class. It differs from the FACS model by the fact that, if shown enough samples, the network can actually learn a general representation of a smile instead of using a fixed and pre-defined structure representation.

The concept of creating general features was explored by Razavian et al. [6] where the first layers of their deep neural network can be reused to achieve good results on similar recognition tasks. Deep neural networks were applied to face expression recognition before, and showed good results. In this area CNNs [7] and **L**ateral **I**nhibition **P**yramidal **N**eural **Net**works (LIPNet) [8] have both been able to produce good results.

The results of such networks raise an interesting question: Is it really possible to achieve good generalizations with the features learned by deep neural networks? To answer this question, we did a study on what kind of knowledge these networks produce. We started implementing a CNN and a LIPNet architecture and trained them with the Cohn-Kanade database [9]. We then applied deconvolution-based processes to see what these networks learned, by visualising which are the maximum responses of the neurons of each network when a face expression is presented. These processes were applied before to CNNs, but we had to implement a novel methodology to visualise the knowledge of LIPNets in a way that we can compare them with the CNN visualisations. Deeper analysis of the visualisations and the comparison of these images with Facial Action Units of the FACS system can show that they correlate [10].

We evaluated our learned features using a data set that was not present during the training step, the Jaffe [11]. This data set allows us to do deeper analysis on the learned features: it contains expressions of female Japanese subjects, which are not present in the Cohn-Kanade database. By doing this experiment, we show that the features learned by these two networks are actually general enough to depict facial features for the unknown images of the Jaffe data set.

In this paper, we discuss the similarity of the learned features of the LIPNet and the FACS model, and the concepts of invariance, redundancy and filtering in the CNN. In Section II, we describe our feature visualisation methods, including our novel LIPNet visualisation strategy. In Section III, we describe our experimental methodology. In Section IV, we exhibit our results and discuss the generalization aspects of the network based on what they learned. Finally, in Section V, we show our conclusions and future work.

II. Feature visualisation methods

CNNs employ the concepts of receptive fields and weight sharing [12]. Through these concepts, the number of trainable parameters is being reduced and the propagation of information through the layers can be calculated by convolution. A signal is convolved with a filter map, containing the shared weights to produce a feature map. As every neuron in the feature map shares its weights, a feature map detects a single feature at all possible input locations. Therefore, it is necessary to have multiple filters and thus multiple feature maps to extract important features.

The LIPNet, in contrast, is a deep pyramidal artificial neural network with lateral inhibition based on the Pyramidal Neural Network (PyraNet) [13], which was motivated by CNNs. The LIPNet incorporates the concepts of lateral inhibition and receptive fields [8]. As CNNs and LIPNets share the concept of receptive fields, they also have similar properties. Consequently, the LIPNet also compresses information throughout its layers. However, as the concept of weight sharing is applied in overlapping regions in their receptive fields only, what each neuron learns is not restricted to simple features such as edges. The neuron connections between each other are more complex. Therefore, the overall number of neurons decreases since it is not necessary to apply multiple filters to capture all features.

Visualising filter maps in CNNs introduced the capability to see what the network learns after training the filters. Several methods have been developed to gain insight into the inner representations of CNNs ([14], [15], [16]). We choose to use the deconvolutional process [16], which gives us an insight of which part of the face expression each neural region of the network activates. Unfortunately, this method is restricted to the architecture of CNNs. We introduce here the concept of visualising features of the LIPNet architecture, based on the deconvolution process. The next sections will describe the deconvolution process and our LIPNet visualisation method.

A. Deconvolutional Networks

Deconvolutional Neural Networks as described by Zeiler et al. [16] are used to visualise a single unit in a feature map of a CNN based on the input. The idea is to visualise maximally activated neurons, as some pixels in the image are responsible for their high activation. To visualise a neuron's activation a in layer l (a^l), the convolution is first carried out as usual. Afterwards, the activation of every neuron in layer l, except for a, is set to the neutral element (usually zero) to visualise a particular neuron.

Subsequently, each step of the convolutional process is reversed. Usually this boils down to *unfiltering*, *unpooling* and an adequate activation function.

Unfiltering is done by reversing the connections. In a Convolutional Neural Network, the Convolutional process can be formally described as

$$v_{nc}^{xy} = \sum_{m} \sum_{h=1}^{H} \sum_{w=1}^{W} w_{(c-1)m}^{hw} v_{(c-1)m}^{(x+h)(y+w)}, \qquad (1)$$

where n represents the feature maps and c the layers, m indexes over the set of feature maps in the (c-1) layer connected to the current layer c. $w_{(c-1)m}^{hw}$ is the weight of the connection between the unit (h,w) within a receptive field connected to the previous layer $c-1$ and to the filter map m. H and W are the height and width of the receptive field.

To reverse the connections, the filters are flipped horizontally and vertically

$$y_{nc}^{xy} = \sum_{m} \sum_{h=1}^{H} \sum_{w=1}^{W} wf_{(c-1)m}^{hw} v_{(c-1)m}^{(x+h)(y+w)}, \tag{2}$$

where $wf_{(c-1)m}^{hw}$ represents the flipped filters and y_{nc}^{xy} describes the generated image.

Unpooling is done by reverting the pooling operation. In our research Max-pooling is used. The Max-pooling can be described as

$$a_j = max_{n \times n} \left(v_{nc} u(x, y) \right), \tag{3}$$

where v_{nc} is the output of the convolution layer. In this function, the Max-pooling computes the maximum activation among the receptive field $u(x, y)$. The maximum operation down-samples the feature map, maintaining the convolution output structure.

The work of Zeiler et al. [16] shows that it is necessary to consider the position of the maximum values, a_j, in order to improve the quality of the visualisations. Therefore, the positions of the maximum values have to be stored in order to recreate the image. Zeiler et al. [16] call these stored position markers `switches`. The other empty positions of the unpooled map are filled with the neutral element regarding the activation function which is usually zero.

The activation functions we use in our convolution layers are **Re**ctified **L**inear **U**nits (ReLU). Although we do not reverse the function, practice has shown that the application of the same function during the deconvolution process yields good visualisations.

Propagating the information contained in a neuron back to input pixel space will produce input neurons with high activation if the original input contains valuable information at that particular location. Due to ReLU being used as the activation function in our CNNs, the neurons learn to participate in the classification through positive values. Zero values on the other hand represent no clues whatsoever, which means that a higher level neuron that depends on its knowledge will also lean towards lower values. Therefore, the final visualisation can be interpreted as follows. Bright pixels indicate high importance of that particular input unit, while dark pixels indicate the opposite. Consequently, in [16] the authors propose to visualise maximally activated neurons and in this way analyse invariance of filter maps. Another option is to visualise every neuron of a feature map and subsequently take the mean of the produced images to analyse the feature map as a whole.

Instead of just visualising a single neuron, it is also possible to visualise every neuron of a feature map and afterwards calculate the mean of these visualisations. Through this procedure we can observe what information is contained in the whole feature map. As features captured by a single neuron might turn out to be very abstract, especially in low layers, this procedure has proven useful to analyse general tendencies of the CNN. A natural consequence of deconvolving each neuron contained in the feature map is that processing times get multiplied by the number of neurons contained in the feature map.

Fig. 1. This image is an illustrative example of the result of the our algorithm. Each cell represents the influence that the input unit causes to get a higher activation value of the selected neuron. The color in these cells is calculated proportionally applying the linear transformation described.

B. Visualising Features in LIPNet

Similarly to the deconvolution process for CNNs, we visualise maximally activated neurons to analyse each pyramidal layer. The reasoning remains the same, as high activation is equivalent to strong participation in the classification. Propagating the signal back to the input pixel space will indicate which parts of the image caused activations in a particular neuron.

As each neuron learns its weights independently, it is necessary to evaluate more than just one or two maximally activated neurons. We combine the visualisations to get a better understanding of what is happening. We do so by taking the weighted arithmetic mean. However, because of the hyperbolic tangent, used as the activation function in the pyramidal layers, values may turn out to be positive, negative or zero. While positive or negative values are equivalent to clues for classification, zero remains the neutral element. To convert input neuron activations into an image, the absolute value is taken as shown in Figure 1. The resulting image can again be interpreted in terms of bright versus dark pixels.

Figure 2 illustrates how a neuron's activation traces back to the input image and thus how features from the neural network's point of view emerge. The idea is based on the gradient ascent algorithm with the objective to maximize the activation of the neuron that is being analysed. The backward propagation of activation is performed through previous layers until the first layer.

III. Methodology

Our goal is to extract the knowledge of the networks and verify if this knowledge contains any generalization capability. To do so, we train the networks with one facial expression data set, the Cohn-Kanade data set, and apply the visualisation techniques to the learned filters. Then, we use the same trained network to recognise face expressions from the Jaffe data set. We then compare the visualisation of the Jaffe corpus with the ones obtained by the Cohn-Kanade corpus.

We do not optimize the architecture of the networks to yield the best recognition rate; instead we build them by

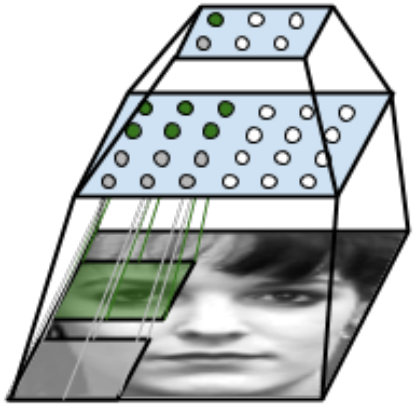

Fig. 2. Given an input image, some neurons in the last layer respond with higher activations (indicated by green colours), while other neurons do not (indicated by grey colours). The cause for higher activations is characterized by combinations of the receptive fields in previous layers which are ultimately based on features in the input image.

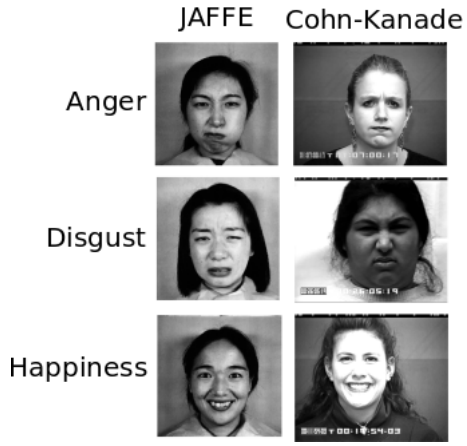

Fig. 3. Examples of face expressions in the Cohn-Kanade and Jaffe datasets.

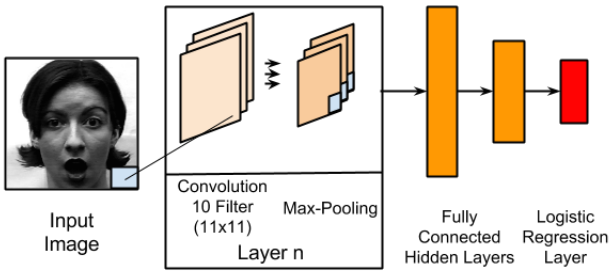

Fig. 4. Architecture of the CNN used for the experiments.

focusing on the quality of the visualisations. We assume that the quality of the visualisation improves when it approaches human facial features. We train our networks several times and visualise the filters of each individual input through the layers. We choose regions where high-level abstraction can be found and correlate the output of these units with the two different datasets.

A. The Dataset

In our experiments we work with the Cohn-Kanade database [17]. The database is composed of 7 different classes that contain sequences of facial expressions which start with a neutral face and end with roughly the apex of the expression. We extracted several images from each sequence. These images represent the same facial expression with different intensities (except for the neutral face). We used the first picture of each sequence to represent an 8th neutral class. The "contempt" class was left out, because it contained few examples, which left us with 7 classes for the Cohn-Kanade database.

We also worked with the Jaffe database for cross-validation. The Jaffe database [11] is composed of seven different classes with one image per expression. The seven classes are congruent with the previously prepared Cohn-Kanade database. Figure 3 exhibits examples of face expressions in these two datasets.

We arranged the images for both databases into three supersets (positive, neutral, negative) according to the emotion annotation and representation language (EARL) [18], which classifies 48 emotions into negative and positive expressions. The images have been cropped to produce quadratic images that approximately centre on the face and have been resized to (128, 128) for CNN and (100, 100) for LIPNet. Finally, in CNN, we applied whitening and subtracted the mean activity over the training set from each pixel, while in LIPNet we equalized the histogram only.

B. The CNN-Architecture

Our experiments revealed that filters tend to be very redundant. We try to keep our networks as sparse as possible, as this facilitates the analysis. Figure 4 shows an illustration of the architecture. We use a standard CNN with 3 convolutional layers in total. The convolutional layers are followed up by two hidden layers. Finally, a logistic regression forms the output. Each convolutional layer has 10 filters with a window size of (11, 11) and is followed by max-pooling with a window size of (2, 2) and a stride of 2. The hidden layers have a size of 500 and 250 neurons, respectively. ReLUs are applied to the convolutional layers, while the hyperbolic tangent function has been applied to the hidden layers.

C. The LIPNet-Architecture

Figure 5 represents the evaluated architecture. It has two pyramidal layers with 30 x 30 neurons and 10 x 10 neurons, respectively; and only one fully connected layer as output. The activation functions are the hyperbolic tangent for the two-dimensional layers and sigmoid for the output layer, which applies a *softmax* function. Table I highlights the basic parameters, such as window size of receptive and inhibitory fields per layer.

IV. Results

We trained our CNNs as well as LIPNets exclusively on the Cohn-Kanade database. Subsequently, we used the JAFFE

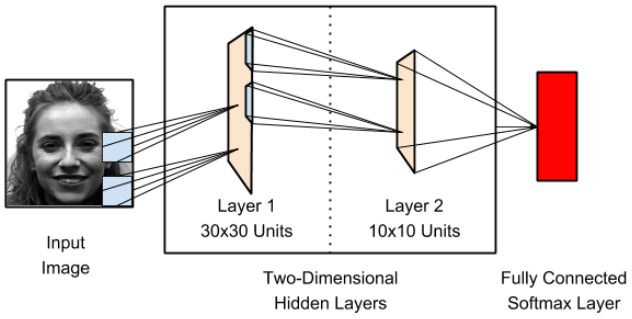

Fig. 5. Architecture of the LIPNet used for the experiments.

TABLE I
LIPNET PARAMETERS: RECEPTIVE FIELD (R), OVERLAP (O), INHIBITORY FIELD (I) AND INHIBITORY WEIGHT (W)

Hidden Layer	Configuration
l=1	(r=15, o=12, i=2, w=0.05)
l=2	(r=4, o=1, i=1, w=0.02)

database for cross-validation. Table II exhibits the recognition rate averages taken from 50 training attempts for the CNN and LIPNet and Cohn-Kanade and JAFFE datasets.

Figure 6 shows visualisations from both architectures. Each column depicts visualisations for one particular input and each row depicts visualisations of different layers. It is possible to see examples of Positive, Neutral and Negative faces for both datasets and which features each network uses to represent each image.

For the CNN we provide visualisations of the second (L2) and the third (L3) layer. We did visualisations of maximally activated neurons as well as the whole feature map. Visualisations of the LIPNet include the first and the second pyramidal 2D layers (responsible for feature extraction) and combined images[1] to emphasize the elements in the input.

We especially use visualisations to improve our architecture. Rather than using a trial-and-error approach, visualisations may guide us to suboptimal performance we can specifically target. Since the learning algorithm itself is subject to abstract parameters, the deficits generally point to architectural issues such as the input size (and through the input size indirectly to the filter size) or the composition of the training set. Neural networks react very sensitive to the composition of the classes. For example, forming natural supersets from classes in an arbitrary database and using these to train a network might worsen the potential results. The reason is that the network needs to know about the differences of the subclasses to tell them apart. Thus, we find our neural networks to produce better results in this case if they are trained on top of a data representation that considers these differences (e.g. conv-layers that have been trained on the subclasses).

Likewise, neural networks react very sensitive to misplaced training examples (in terms of the class they've been put in). For example, in case of poor cross-validation results, some might be quick to judge it as a consequence of overfitting. There is a difference, though, between filters for instance that do not detect anything and filters that do detect something. Although the network's performance is worse on the JAFFE database, it is aware of the fact that faces are being displayed. As such, the facial expression of the databases might differ, causing confusion.

[1]The original input combined with the image of the visualised neurons.

TABLE II
RECOGNITION RATES. AVERAGES WERE TAKEN FROM 50 TRAINING ATTEMPTS.

%	Cohn-Kanade		JAFFE	
Accuracy	Best	Avg.	Best	Avg.
CNN	87.0	85.1	49.0	47.3
LIPNet	75.0	62.4	61.0	37.4

A. Features Learned by CNNs

As mentioned above we used visualisations to improve the architecture of our neural networks. Flaws in the images, such as *focusing the wrong features* or *blurred facial features* among others, can be revealed and adequately treated by understanding how filters cooperate with each other. By doing this, recognition rates will also improve as our experiments empirically indicate. The filters in CNNs split the work between each other, which results in a hierarchical structure. This structure is subject to three properties in particular:

- Invariance
- Redundancy
- Filtering

The **invariance** of a filter is clearly visible through visualisations of the respective feature map. Each filter specialises in a particular feature which it can reliably detect. Figure 7 shows visualisations of a filter with a focus around the eyes. We inserted one image that shows that the filters cannot detect the same features, but in a well trained CNN, the filters will mostly display the same feature. Note also that examples of particular classes might pose a problem, but generally we observed that the feature is being extracted independently of the input class. Besides, the filters' inability to extract a particular feature for all given inputs might be a cause for redundancy.

The **redundancy** of filters is expressed through visualisations which focus on the same features. As filters cannot detect the same feature for every given input, redundancy is necessary to cover these filters. A strong presence of a feature might also indirectly influence the classification process. Another reason for redundancy is the use of too many filters which will naturally result in similar features being detected as the neural network has enough trainable parameters at its disposal. Figure 8 shows visualisations of different filters for the same input.

The **filtering** is characterised by the complexity of the features detected by a filter map. Each filter map combines the features of the previous layer to produce a more complex output. Because the captured features get more complex, the feature map as a whole will carry less information as the

features become more unique, as for example nose, mouth, etc. on higher layers and edges, arks, etc. on lower layers.

We conclude that convolutional layers do not directly take part in the classification process. Rather do they provide a convenient data representation, suited to the task, for the subsequent multilayer perceptron to operate on. The filters split the work between each other. Through redundancy the availability of important features is assured. With each layer unimportant information and noise is filtered out which leaves an easily classifiable data representation of the initial input in terms of the task at hand.

The quality of the visualisations is a good indicator of the networks performance in terms of accuracy and can be used as a means to improve the neural network. For CNNs in particular, we observed that the more the feature resembles the object displayed in the original input, the smaller the classification error is. If the features seem to be good, but the image is classified incorrectly anyway, the images of the database might be labelled incorrectly. Flaws of the image such as *focusing on the wrong features* or *blurred facial features* can point to particular architectural problems, such as using supersets instead of the subclasses for training which will prevent the CNN from learning the differences between the subclasses.

B. Features Learned by LIPNets

In general, the features extracted in these images are similar. One reason for this behavior is that all images are aligned and do not present interferences of background. Therefore, it

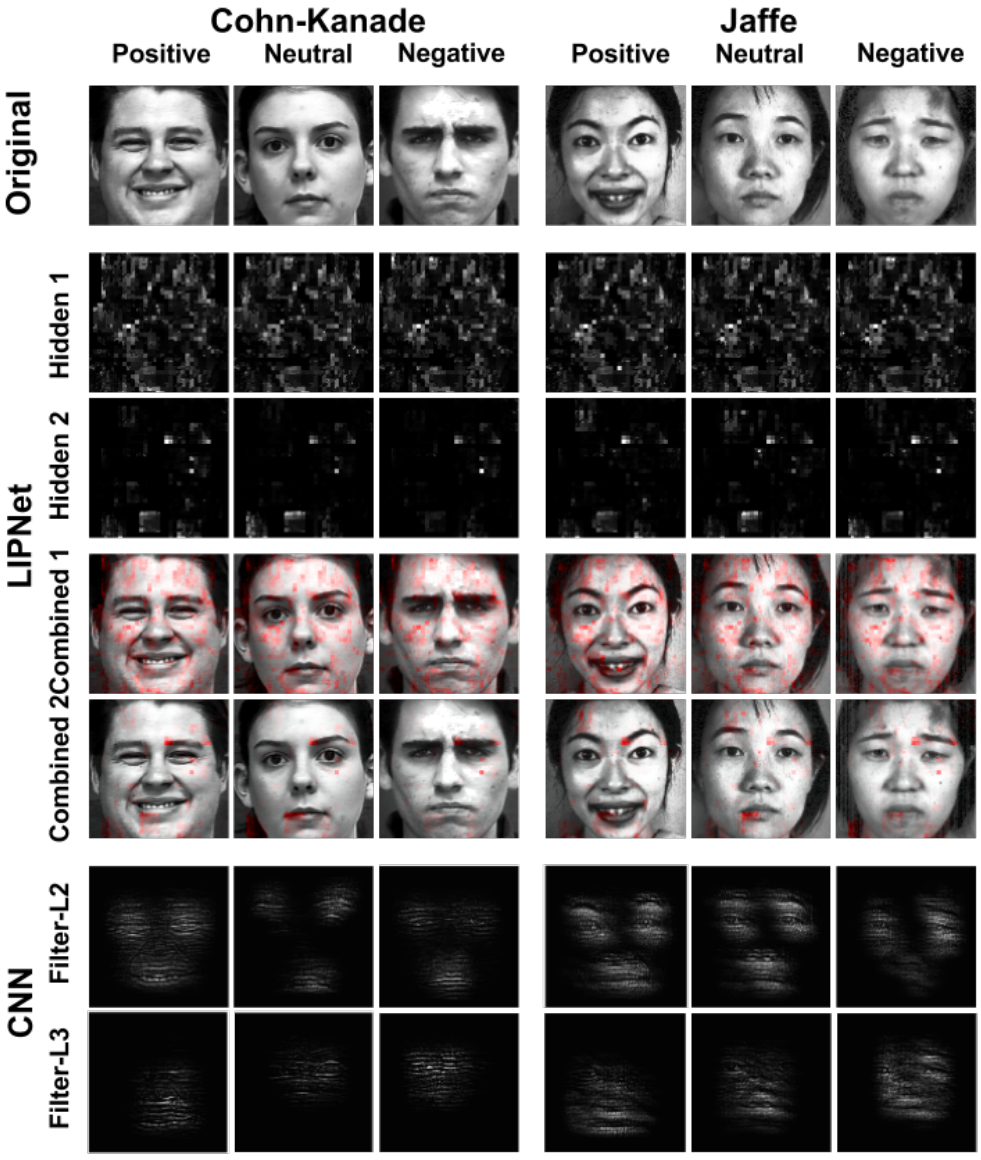

Fig. 6. Visualisations for the Cohn-Kanade and the Jaffe database with both architectures. Row 1 shows the original image. Rows 2 and 3 show visualisations of the first and second pyramidal layers of the LIPNet. The images have been combined with the original image to emphasize the filtered elements in the rows 4 and 5. Rows 6 and 7 show visualisations of the second and third convolutional layer of the CNN.

Fig. 7. This figure shows the invariance of a filter. Each visualisation was taken from the same feature map with a different example. Row one shows the input while row two shows the respective visualisation.

is important to notice that the first hidden layer in LIPNet looks for features in all images, disregarding some parts, such as hair and background where the pixels are closer to black. To the contrary, the whiter pixels are concentrated in regions alike the FACS, in Upper Face Action Units and Lower Face Action Units. The third image in Figure 9 was processed to remove the areas with a low contribution to a high activation value of the neurons. We can observe areas that correspond to the activation of the muscles: *Frontalis*, *Zygomaticus*, *Levator Labii Superioris Alaquae Nasi*, *Mentalis* and *Depressor Labii Inferioris*. These muscles have a great correlation with emotion classification.

The features learned by the second hidden layer are considered as key parts only from the features in the previous layer. So, we can see in row four of Figure 6 images cleaner than in row two. Furthermore, the structure of features that correlates with action units is presented around the eyes, forehead, nose and mouth [19].

Although the features look similar, it is important to notice that when a particular pattern of muscle action appears in the face, the receptive field stays more active showing patterns like outer brow raiser, lid tightener, nose wrinkler, lip corner puller, lower lip depressor, etc. We can observe this difference between column one and three, both in row five. In the positive, the regions that comprise the cheeks and mouth are more active than in the negative example, turning these regions darker when the *Zygomaticus* muscle is active while raising the cheeks. This phenomenon is coded by AU 12, typical feature of positive label, when the person is smiling. Another evident learned feature is the region around the eye presented in this subset. It is important to characterize almost all upper face action units, especially the brow region, which is the most

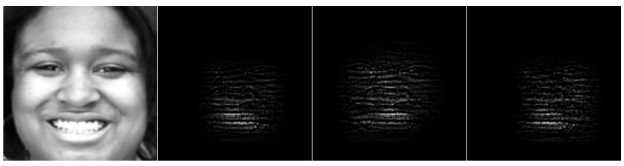

Fig. 8. This figure shows how redundancy affects filters of a layer. Each visualisation was taken from a different filter but with the same input.

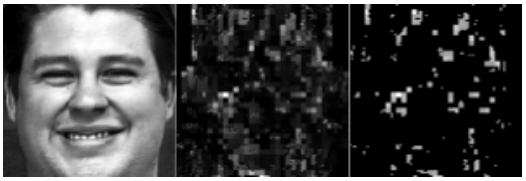

Fig. 9. On the left, the original image; the image in the middle is the features learned by the first hidden layer from LIPNet; and the last image is the middle image processed to show the areas that contribute more for a higher activation. All pixels less than 80 intensity, or 30% of the contribution, were removed from the image.

Fig. 10. Illustration of the evolution of a positive expression. The first row is a start stage that has been classified as negative label, while the others, until the end expression, have been classified as positive label.

active one. So, the LIPNet analyses the AUs 1, 2, 4, 5 and 7, which code for raised inner brow, raised outer brow, lowered v, raised upper lid, and tightened lid, respectively. The same occur in the Jaffe data set in Figure 6, the LIPNet looks for similar features.

Finally, Figure 10 shows the evolution of a positive expression; being the initial stage in the first row, and the apex in the last. We can observe in column two that the important areas for classification remain the same, but the difference between the expressions in each row increases within these areas: the eyes become more expressive and the mouth becomes increasingly open, respectively. As the features become more evident, the probability to be in a positive class increases. In the first row, the LIPNet classified the image with a negative label, whereas for the other image, the LIPNet classified the image with a positive label and the probability becomes greater for the following rows: 38%, 45%, and 48%.

V. Conclusions

Both architectures, CNNs as well as LIPNets, are able to cope with the datasets while concentrating on different aspects of the inputs as illustrated in Figure 6. The distinct visualisations stem from the architectural differences. As CNNs employ weight sharing, a filter map learns to detect a particular feature in all locations of the input, whereas neurons in pyramidal layers are restricted to their receptive field. Thus, they have to learn all possible variations that might appear in their particular receptive field. As a consequence, individual input neurons or rather the smaller receptive fields have a stronger effect on the activation of neurons in subsequent layers, which is clearly visible in Figure 6. Cropping the images to remove background disturbances or the absence of positional issues (all subjects are portrayed in the same way) might have amplified the effect. Naturally, by using an adequate amount of layers the receptive field gets larger throughout the layers and, consequently, the impact of individual neurons weakens. Our approach to use rather sparse architectures might have also impeded the results of LIPNets.

When we are trying to understand CNNs through visualisations, what we are actually doing is trying to understand the data it operates on. As such, visualisations may not provide absolute statements about CNN architectures in general but rather specific clues for the task at hand. However, we can use these clues to improve our understanding of the task and consequently improve the architecture.

In our experiments, we showed how the CNNs and LIPNets learn facial structures and that the learned representations are reliable and can be extracted from different images with different faces. To do so, we use the deconvolution process to visualise CNN features and we introduce a novel algorithm to visualise LIPNet features. We used the visualisations to demonstrate concepts such as invariance, redundancy and filtering, which are common for deep neural networks. We discussed how each network learned different representations, and how the structure and processing of each network affected the learned facial structures. The features learned by the LIPNet are comparable to the Facial Action Units - FACS model, and we show how this network could generalize these features in different images.

We want to use the knowledge obtained by these experiments to improve further the models, and use them as a tool to help with the modification of the architectures, fine-tune the parameters and build optimal training strategies. Also, we want to extend the observation of the emotion expressions by using sequence information and comparing it to human perception patterns.

Acknowledgment

This work was partially supported by the CAPES Brazilian Federal Agency for the Support and Evaluation of Graduate Education (p.n.5951–13–5), the German Research Foundation DFG under project CML (TRR 169), and the Hamburg Landesforschungsförderungsprojekt.

References

[1] P. Ekman, "Facial expression and emotion." *American psychologist*, vol. 48, no. 4, p. 384, 1993.

[2] P. Ekman and W. Friesen, *Facial Action Coding System: A Technique for the Measurement of Facial Movement.* Palo Alto: Consulting Psychologists Press, 1978.

[3] T. J. Balkin, W. J. Horrey, R. C. Graeber, C. A. Czeisler, and D. F. Dinges, "The challenges and opportunities of technological approaches to fatigue management," *Accident Analysis & Prevention*, vol. 43, no. 2, pp. 565 – 572, 2011, advancing Fatigue and Safety Research.

[4] D. F. Dinges, S. Venkataraman, E. L. McGlinchey, and D. N. Metaxas, "Monitoring of facial stress during space flight: Optical computer recognition combining discriminative and generative methods," *Acta Astronautica*, vol. 60, no. 4, pp. 341–350, 2007.

[5] M. Valstar and M. Pantic, "Fully automatic facial action unit detection and temporal analysis," in *Proceedings of the 2006 Conference on Computer Vision and Pattern Recognition Workshop*, ser. CVPRW '06. Washington, DC, USA: IEEE Computer Society, 2006, pp. 149–.

[6] A. S. Razavian, H. Azizpour, J. Sullivan, and S. Carlsson, "CNN features off-the-shelf: an astounding baseline for recognition," *CoRR*, vol. abs/1403.6382, 2014.

[7] D. Hamester, P. Barros, and S. Wermter, "Face expression recognition with a 2-channel convolutional neural network," in *Proceedings of International Joint Conference on Neural Networks (IJCNN)*, Killarney, Ireland, 2015, pp. 1787–1794.

[8] B. Torres Fernandes, G. Cavalcanti, and T. I. Ren, "Lateral inhibition pyramidal neural network for image classification," *Cybernetics, IEEE Transactions on*, vol. 43, no. 6, pp. 2082–2092, Dec 2013.

[9] 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*, ser. CVPRW '06. Grenoble, France: IEEE Computer Society, 2000, pp. 46–53.

[10] P. Khorrami, T. L. Paine, and T. S. Huang, "Do deep neural networks learn facial action units when doing expression recognition?" *CoRR*, vol. abs/1510.02969, 2015. [Online]. Available: http://arxiv.org/abs/1510.02969

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

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

[13] S. L. Phung and A. Bouzerdoum, "A pyramidal neural network for visual pattern recognition," *Trans. Neur. Netw.*, vol. 18, no. 2, pp. 329–343, Mar. 2007.

[14] K. Simonyan, A. Vedaldi, and A. Zisserman, "Deep inside convolutional networks: Visualising image classification models and saliency maps," *CoRR*, vol. abs/1312.6034, 2013.

[15] D. Erhan, Y. Bengio, A. Courville, and P. Vincent, "Visualizing higher-layer features of a deep network," University of Montreal, Tech. Rep. 1341, Jun. 2009.

[16] M. D. Zeiler and R. Fergus, "Visualizing and understanding convolutional networks," *CoRR*, vol. abs/1311.2901, 2013.

[17] 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*, ser. FG '00, Grenoble, France, 2000, pp. 46–53.

[18] M. Schröder, H. Pirker, M. Lamolle, F. Burkhardt, C. Peter, and E. Zovato, "Representing Emotions and Related States in Technological Systems," in *Emotion-Oriented Systems*, ser. Cognitive Technologies, R. Cowie, C. Pelachaud, and P. Petta, Eds. Springer Berlin Heidelberg, 2011, pp. 369–387.

[19] Y.-L. Tian, T. Kanade, and J. Cohn, "Facial expression analysis," in *Handbook of Face Recognition*. Springer New York, 2005, pp. 247–275.