Can N-dimensional Convolutional Neural Networks Distinguish Men And Women Better Than Humans Do?

Iveta Mrázová, Josef Pihera and Jana Velemínská

***Abstract*— Convolutional neural networks (CNN) were able to beat human performance in various areas of 2D image recognition, e.g., in the German Traffic Sign competition run by IJCNN 2011. While the majority of classical image processing techniques is based on carefully pre-selected image features, CNNs are designed to learn local features autonomously. A growing availability of high-dimensional object data, e.g., from medicine or forensic analysis, thus motivated us to develop a new variant of the classical CNN model.**

The introduced N-dimensional convolutional neural networks (ND-CNN) enhanced with an enforced internal knowledge representation allow to process general N-dimensional object data while supporting adequate interpretation of the found object characteristics. Experimental results obtained so far for gender classification of 3D face scans confirm an extremely strong power of the proposed neural classifier. The developed ND-CNNs significantly outperformed humans (by 33%) while still allowing for a transparent representation of the face features present and detected in the data.

I. Introduction

CONVOLUTIONAL neural networks (CNN) provide an extremely powerful means designed for 2D-image classification. Classical image recognition systems often consist of a customized task-specific feature detector and a general-purpose and trainable classifier. Yet CNNs allow to process any type of images due to automatic pre-processing. In this respect, a notable interest awoke the German Traffic Sign competition run last year by the IJCNN 2011 conference where convolutional neural networks achieved a better-than-human accuracy of 99.15% [1], [13].

Quite recently, also the amount of available 3D-data and their usage grow immensely. In perinatal medicine, e.g., [2] demonstrates a divergence in craniofacial structure when comparing fetuses with Down's syndrome to the healthy ones. In forensic science, facial recognition from 3D data is widely used for identity verification [4]. Other areas include evolution (facial reconstruction of our ancestors), orthodontics, plastic surgery, etc.

Obviously, sexual dimorphism is responsible for a substantial part of human facial variability. In pubertal males, a high testosterone-to-oestrogen ratio causes an overall elongation of the face, a larger nose, the lateral growth of the cheekbones, and the lengthening of the lower face, giving the face a more robust shape. The influence of oestrogen leads on the other hand to a more gracile and rounded shape of the face with high eyebrows, a more gracile nose and chin and fuller lips [14].

Iveta Mrazova and Josef Pihera are with the Department of Theoretical Computer Science and Mathematical Logic, Faculty of Mathematics and Physics, Charles University in Prague, Czech Republic (email: iveta.mrazova@mff.cuni.cz).

Josef Pihera is also with the Database and Artificial Intelligence Group, Vienna University of Technology, Austria.

Jana Veleminska is with the Department of Anthropology and Human Genetics, Faculty of Science, Charles University in Prague, Czech Republic.

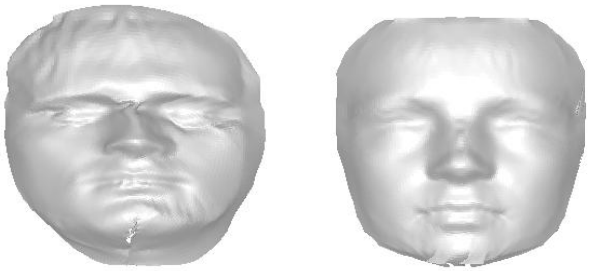

Fig. 1. Normalized artificially generated face model of a man (left) and of a woman (right) seen from the front.

Traditional CNNs do not require advanced data pre-processing while providing generally superior classification rates. Therefore, we will explore here the possibility to analyze also more-dimensional object data with CNN-like networks, i.e. roughly pre-processed 3D facial scans in our case. For an example of such data, see Fig. 1. The processed data contained scanned faces of real people. For privacy reasons, we were, however, allowed to use only artificially created 3D face models (that contain a few apparent irregularities) for illustrations in this paper. Anyway, the performed experiments involved all the provided real face scans.

The following section analyzes neural network paradigms related to CNNs and an adequate interpretation of the formed network structure. The next section introduces the new model of N-dimensional convolutional neural networks (ND-CNN) enhanced with enforced internal knowledge representation. Afterwards, the results of supporting experiments (involving gender classification of 3D-face models) will be discussed. The concluding section summarizes the achieved results.

A. Human performance

At the first glance, gender classification might seem to be simple, at least from the human perspective. Yet, we are using many secondary hints like body shape, hair and clothes. To understand how difficult the task is, 21 persons participated in experiment where they had to identify the gender of persons whose face shape was shown in a questionnaire. There were 60 faces selected randomly from a data set containing 122 faces (33 of them were female faces and 27 male faces). The results can be seen in Table I. The success rates were

TABLE I
SUCCESS RATES OF HUMAN PARTICIPANTS ON THE FACE MODELS

Gender of the participants	***Gender of the faces***		
	Men	Women	All
Men	**78.1%**	**54.5%**	**65.2%**
Women	76.8%	51.0%	62.6%
All	77.4%	52.7%	63.8%

calculated as the number of correct answers (true positives) divided by the number of the considered faces.

Further, there appeared some interesting facts about the results of humans. The participants more often labeled the faces as male, see Table I. 3 male models were labeled correctly by all participants while the highest rate of one female model was 95.2%. Gender classification thus might be a relatively challenging task for humans without the field expertise of an anthropologist.

II. RELATED WORKS

The original CNN model uses a slightly modified version of the well-known back-propagation algorithm (BP) for training. To keep the paper self-contained, we will briefly discuss also the BP-algorithm for fully-connected feed-forward neural networks (BP-networks). For the BP-training algorithm, the training set T is a finite non-empty set of P ordered pairs of input/output patterns: $T = \{ [\vec{x}_1 , \vec{d}_1], \ldots , [\vec{x}_P , \vec{d}_P] \}$. A neuron with the weights $(w_1, \ldots, w_n)$, the threshold ϑ and the input vector $\vec{z} = (z_1, \ldots, z_n)$, computes its potential ξ as $\xi = \sum_{i=0}^{n} z_i w_i$, where $w_0 = \vartheta$ and $z_0 = 1$. For the neurons, we will consider the hyperbolic tangent transfer function:

$$y = f(\xi) = \frac{1 - e^{-2\xi}}{1 + e^{-2\xi}} \quad , \tag{1}$$

Its derivative $f'(\xi)$ equals to:

$$f'(\xi) = (1 + y)(1 - y) = 1 - y^2 . \tag{2}$$

In BP-networks, the neurons are grouped into $L+1$ mutually disjunctive sets called layers. The L-th layer consists of m_L output neurons, while the 0-th layer contains m_0 input neurons, formally setting its output $\vec{y}^0 = (y_1^0, \ldots, y_{m_0}^0)$ to the presented input pattern $\vec{x} = x_1, \ldots, x_{m_0}$: $\vec{y}^0 = \vec{x}$.

The aim of the standard BP-training algorithm is to find a set of weights that ensure that for each input pattern the actual output produced by the network is the same as (or sufficiently close to) the output pattern. The desired behavior is evaluated by the objective function E:

$$E = \frac{1}{2} \sum_{p=1}^{P} \sum_{j=1}^{m_L} (y_j^{L,p} - d_j^p)^2 \tag{3}$$

where p is an index over all P training patterns, j is an index over all output neurons, y is their actual and d is their desired output value. Omitting the index p for the desired and actual neuron output values d and y, the weights $w_i^{j,l}$ between the neurons from the $(l-1)$-th layer denoted by i and the neurons from the l-th layer indexed by j of the network are adjusted iteratively after presenting each respective training pattern according to $-\partial E / \partial w_i^{j,l}$ by:

$$w_i^{j,l}(t+1) = w_i^{j,l}(t) + \alpha \delta_j^l y_i^{l-1} \tag{4}$$

The terms for δ_j^l correspond to:

$$\delta_j^l = \begin{cases} (d_j - y_j^l)(1 + y_j^l)(1 - y_j^l) \\ \qquad \text{for an output neuron } (l = L) \\ (1 + y_j^l)(1 - y_j^l) \sum_{k=1}^{m_{l+1}} \delta_k^{l+1} w_j^{k,l+1} \\ \qquad \text{for a hidden neuron } (l < L) \end{cases} \tag{5}$$

i, j and k index the neurons in the respective layer $l; 1 \leq l \leq L$. m_l denotes the number of neurons in the layer l. $t+1$ and t index next and present weights, respectively, α stands for the learning rate.

A. Learning internal representation

In order to clarify the role of hidden neurons, the method enforcing the so-called condensed internal representation [11] groups the outputs of the hidden neurons around three possible values 1, -1 and 0 corresponding to active, passive and to the so-called silent states. In the AI-terminology, this corresponds to creating rules of an expert system, where active neuron states indicate "yes", passive states "no" and the so-called silent states stand for those cases where "no decision is possible." Without the loss of generality, for the hidden layer $l^*(1 \leq l^* \leq L)$, the criterion for developing a condensed internal representation will be formulated as an additional term of the objective function to be minimized during training:

$$F = \sum_{p=1}^{P} \sum_{h=1}^{m_{l^*}} (1 + y_h^{l^*,p})^s (1 - y_h^{l^*,p})^s (y_h^{l^*,p})^2 \tag{6}$$

where p is an index over all P training patterns and h an index over all hidden neurons. m_l^* denotes the number of neurons in the layer l^*. y represents their output value. The respective terms achieve their minima for one of the values 1, -1 and 0. s tunes the shape of the representation error function F (a recommended value is $s = 4$). Now, the new objective function has the form:

$$G = E + c_F F \tag{7}$$

where E represents the standard BP-error function and F stands for the above defined representation error function. c_F reflects the trade-off between the influence of E and F in G. To minimize G, gradient descent according to

$$\Delta_G w_i^{j,l} \cong -\frac{\partial G}{\partial w_i^{j,l}} = -\left(\frac{\partial E}{\partial w_i^{j,l}} + c_F \frac{\partial F}{\partial w_i^{j,l}} \right) \tag{8}$$

can be applied as well by adjusting the weights in the following way:

$$w_i^{j,l}(t+1) = w_i^{j,l}(t) + \alpha \delta_j^l y_i^{l-1} + \alpha_r \varrho_j^l y_i^{l-1} \tag{9}$$

In this formula, δ_j^l is defined by the term (5) and

$$\varrho_j^l = \begin{cases} 0 \quad \text{for a hidden or output neuron } l > l^* \\ 2\left[(s+1)(y_j^l)^2 - 1\right](1+y_j^l)^s(1-y_j^l)^s y_j^l \\ \quad \text{for a hidden neuron from layer } l = l^* \\ (1+y_j^l)(1-y_j^l)\sum_{k=1}^{m_{l+1}} \varrho_k^{l+1} w_j^{k,l+1} \\ \quad \text{for a hidden neuron from layer } l < l^* \end{cases} \tag{10}$$

w stands for the weights. i indexes the neuron connected with neuron j via the weight w_{ij}. y_j is the actual output value of the neuron j. s tunes the shape of the representation error function. $t+1$ and t index next and present weights, respectively. α and α_r represent the particular learning rates.

B. Convolutional neural networks

CNNs [8] are known to outperform all other paradigms when used for 2D-image recognition with minimum or no advanced pre-processing. The principles of weight sharing and local receptive fields keep down the number of trainable parameters in CNNs. With biologically inspired local receptive fields [5], perceptron-like neurons can extract elementary visual features such as oriented edges or corners. The extracted features are then combined by the subsequent layers in order to detect more complex higher-order features. In each layer, the neurons are grouped in the so-called feature maps. Once a feature is detected by the feature-extracting layer, a layer of sub-sampling neurons performs local averaging to blur the exact position of the feature. Successive convolutional and sub-sampling layers are typically alternated. In CNNs, however, at each layer, the number of feature maps is increased while their spatial resolution is decreased. Neurons from higher layers may have input connections from several feature maps in the preceding sub-sampling layer. Finally, CNNs contain two layers of perceptron-like neurons representing a classifier of the outputs received from the last sub-sampling layer.

LeNet-5 is a CNN consisting of 7 layers (not counting the input) that can be trained e.g. by means of back-propagation [8]. Its structure, number and size of feature maps used in the respective layers, is shown in Fig. 2. The overlapping perception windows in the first, third and fifth layer are of the size 5×5. All the neurons in the respective feature map thus share the same set of 25 weights and the same bias detecting the same feature at all possible locations in the input. In the remaining layers, the size of the non-overlapping perception windows is 2×2. In the sub-sampling layers, the four input values of each neuron are summed together, multiplied by an adjustable coefficient, added to an adjustable bias and passed through a sigmoidal activation function. The sixth layer contains 84 fully connected perceptron-like neurons with one output unit per class in the recognition task.

III. N-Dimensional CNN- networks with enforced internal representation (ND-CNN)

The original CNN model was designed to process 2D-images and cannot deal with general N-dimensional object data. On the other hand, a CNN model supporting also a higher dimensionality would allow for many further applications. A three-dimensional CNN was used in [6] to process time sequences of 2D images (video). For $N = 4$, we could, e.g., model the motion of 3D objects in time or extrapolate, how the shape of a face is changing with age. A similar technique could be also applied to 4D sonograms which already represent an important tool for clinical praxis [7]. An example for 5-dimensional CNNs might be processing of radio astronomic data; the 5th dimension would then characterize the measured data in various wave lengths [15]. In such a case, one 5D voxel could represent a part of a gas cloud in time t in the Roentgen spectrum, while another voxel would represent the same part of the gas cloud in time t but in the infra-red spectrum.

From this point of view, we will introduce now the model of general N-dimensional convolutional neural networks further denoted as ND-CNN. Higher-dimensional input data, we will be presented to the network in the form of an Nth-order tensor of values $y_{\mathbf{z}}^{0,0}$, where the N-dimensional index $\mathbf{z}$ corresponds to an N-tuple $\mathbf{z} = (z_1, \ldots, z_N)$. In order to conform to the high-dimensional structure of the data, each feature map of both the convolutional and sub-sampling layer l will have the shape $m_1^l \times m_2^l \times \ldots \times m_N^l$, where m_d^l is the size of all feature maps of the layer l in dimension d. Each layer l thus comprises a set F_l of feature maps that contain altogether $|F_l| \cdot \prod_{d=1}^{N} m_d^l$ neurons. $y_{\mathbf{z}}^{f,l}$ denotes the output of a neuron $(z_1, z_2, \ldots, z_N, f, l)$, situated at the position $\mathbf{z} = (z_1, \ldots, z_N)$ of an N-dimensional feature map f from the layer l, where $0 \le z_d < m_d^l$. As a shorthand for $(z_1, z_2, \ldots, z_N, f, l)$, we will write $(\mathbf{z}, f, l)$.

Feed-forward layers, however, do not depend on the internal structure of convolutional or sub-sampling layers. Assume that the layer $l-1$ is a convolutional or a sub-sampling layer. If, for example, the layer l is an ordinary feed-forward layer, its neurons are connected to all neurons from the layer $l-1$ without considering their organization inside the layer $l-1$.

A. The Convolutional Layer

Receptive fields of the convolutional layer l have now the size $(r_c^l)^N$. The neuron $(\mathbf{z}, f, l)$ is thus connected to all neurons $(z_1 + \triangle z_1, \ldots, z_N + \triangle z_N, f', l-1)$ with $f' \in F_{f,l}^{IN}, 0 \le \triangle z_d < r_c^l, 1 \le d \le N$. Denoting $\triangle \mathbf{z} = (\triangle z_1, \ldots, \triangle z_N)$, we can use the shorthand $(\mathbf{z} + \triangle \mathbf{z}, f', l-1)$ for $(z_1 + \triangle z_1, \ldots, z_N + \triangle z_N, f', l-1)$. Further, we will limit our considerations receptive fields shifted by exactly one neuron in the respective dimension.

A weight $w_{\triangle \mathbf{z}}^{f,f',l}$ connects thus every neuron $(\mathbf{z}, f, l)$ from the l-th layer with the neuron $(\mathbf{z} + \triangle \mathbf{z}, f', l-1)$ from the $(l-1)$-th layer for all applicable $\triangle \mathbf{z}$. This weight is shared by all the neurons from the same feature map located at any

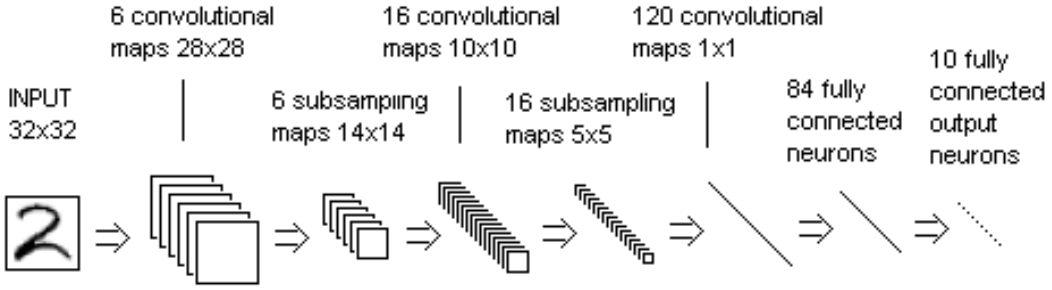

Fig. 2. The structure of the LeNet-5 convolutional neural network

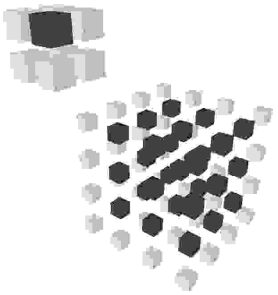

Fig. 3. Convolution of 4x4x4 feature map (right) with 3x3x3 receptive field and 3 dimensions.

position $\mathbf{z}$. The potential $\xi_{\mathbf{z}}^{f,l}$ corresponding to the output $y_{\mathbf{z}}^{f,l}$ of the neuron $(\mathbf{z}, f, l)$ is defined as:

$$y_{\mathbf{z}}^{f,l} = \xi_{\mathbf{z}}^{f,l} = \sum_{f' \in F_{f,l}^{IN}} \sum_{\triangle \mathbf{z} \in \{0,\ldots,r_c^l - 1\}^N} y_{\mathbf{z}+\triangle \mathbf{z}}^{f',l-1} \cdot w_{\triangle \mathbf{z}}^{f,f',l} \quad (11)$$

The size $m_1^l \times m_2^l \times \ldots \times m_N^l$ of all feature maps in the convolutional layer l is again imposed by the size of the feature maps in layer $l-1$ and by the size of the receptive field $\prod_{d=1}^N r_c^l$. We obtain thus $m_d^l = m_d^{l-1} - r_c^l + 1$ for all $1 \leq d \leq N$.

B. The sub-sampling layer

In the ND-CNN model, the sub-sampling layers reduce the size of the feature maps in all N dimensions. As we do not suppose here any overlapping of the sub-sampled areas, each neuron from layer $l-1$ is thus connected to exactly one neuron in layer l. The size of the feature maps will then be changed according to the amount of sub-sampling $r_s^{l,1}, \ldots, r_s^{l,N}$. Given the size $m_1^{l-1} \times \ldots \times m_N^{l-1}$ of the layer $l-1$, the size of the layer l after sub-sampling will correspond to $m_d^l = m_d^{l-1}/r_s^{l,d}$ for all $1 \leq d \leq N$. In order to determine the output of a sub-sampling layer, we will first define an auxiliary set S_l and an auxiliary function g:

$$S_l = \prod_{d=1}^N \{0, \ldots, r_s^{l,d} - 1\} \quad (12)$$

$$\begin{aligned} g(\mathbf{z}) &= g(z_1, z_2, \ldots, z_N) = \\ &= (z_1 \cdot r_s^{l,1}, z_2 \cdot r_s^{l,2}, \ldots, z_N \cdot r_s^{l,N}) \end{aligned} \quad (13)$$

For the neuron $(\mathbf{z}, f, l)$ in the sub-sampling layer l we determine the potential $\xi_{\mathbf{z}}^{f,l}$ in the case of averaging according to:

$$\xi_{\mathbf{z}}^{f,l} = \frac{1}{\prod_{d=1}^N r_s^{l,d}} \sum_{\mathbf{z} \in S_l} y_{g(\mathbf{z})+\triangle \mathbf{z}}^{f,l-1} , \quad (14)$$

and in the case of maximization as:

$$\xi_{\mathbf{z}}^{f,l} = \max_{\triangle \mathbf{z} \in S_l} \left(y_{g(\mathbf{z})+\triangle \mathbf{z}}^{f,l-1} \right). \quad (15)$$

The output $y_{\mathbf{z}}^{f,l}$ corresponds then (for some parameters $a^{f,l}$ and $b^{f,l}$) to:

$$y_{\mathbf{z}}^{f,l} = f \left(a^{f,l} \cdot \xi_{\mathbf{z}}^{f,l} + b^{f,l} \right). \quad (16)$$

C. The Learning Process

The architectural modifications of the ND-CNN model also imply changes in the computation of the error gradient. Otherwise the learning process remains unchanged. Essentially, just the sums over the feature maps and over the receptive field must conform to the increased dimensionality. Further, for the neuron j from the l-th layer, the term $\partial G/\partial y_j^l = \partial E/\partial y_j^l + c_F \cdot \partial F/\partial y_j^l$ can be determined by means of (17) – (18):

$$\frac{\partial E}{\partial y_j^l} = \begin{cases} y_j^l - d_j & \text{for an output neuron } (l = L) \\ \sum_{k=1}^{m_{l+1}} \frac{\partial E}{\partial \xi_k^{l+1}} w_j^{k,l+1} & \\ & \text{for other neurons } (l < L) \end{cases} \quad (17)$$

$$\frac{\partial F}{\partial y_j^l} = \begin{cases} 0 & \text{for a neuron from layer } l > l^* \\ 2\left[1 - (s+1)(y_j^l)^2\right](1 - (y_j^l)^2)^{s-1} y_j^l & \\ & \text{for a neuron from layer } l = l^* \\ & \text{(also the } l^*\text{-th convolutional layer)} \\ (1 + y_j^l)(1 - y_j^l) \sum_{k=1}^{m_{l+1}} \frac{\partial F}{\partial \xi_k^{l+1}} w_j^{k,l+1} & \\ & \text{for a neuron from layer } l < l^* \end{cases} \quad (18)$$

1) The sub-sampling layer: We have to consider just the terms $\partial G/\partial a^{f,l}$ and $\partial G/\partial b^{f,l}$ in the sub-sampling layer l. Applying the chain rule and using (16), we obtain:

$$\frac{\partial G}{\partial a^{f,l}} = \sum_{\mathbf{z}} \frac{\partial G}{\partial y_{\mathbf{z}}^{f,l}} \cdot f'(a^{f,l} \cdot \xi_{\mathbf{z}}^{f,l} + b^{f,l}) \cdot \xi_{\mathbf{z}}^{f,l} \quad (19)$$

$$\frac{\partial G}{\partial b^{f,l}} = \sum_{\mathbf{z}} \frac{\partial G}{\partial y_{\mathbf{z}}^{f,l}} \cdot f'(a^{f,l} \cdot \xi_{\mathbf{z}}^{f,l} + b^{f,l}) \cdot 1 \quad (20)$$

To continue with gradient calculations backwards for the layer $l-1$, we need the derivative $\partial G/\partial y_{\mathbf{z}'}^{f',l-1}$. Again, we start with the chain rule:

$$\frac{\partial G}{\partial y_{\mathbf{z}'}^{f',l-1}} = \frac{\partial G}{\partial y_{\mathbf{z}}^{f,l}} \cdot \frac{\partial y_{\mathbf{z}}^{f,l}}{\partial \xi_{\mathbf{z}}^{f,l}} \cdot \frac{\partial \xi_{\mathbf{z}}^{f,l}}{\partial y_{\mathbf{z}'}^{f',l-1}} \quad (21)$$

The derivative $\partial G/\partial y_{\mathbf{z}}^{f,l}$ is assumed to be known. $\partial y_{\mathbf{z}}^{f,l}/\partial \xi_{\mathbf{z}}^{f,l}$ follows from (16):

$$\frac{\partial y_{\mathbf{z}}^{f,l}}{\partial \xi_{\mathbf{z}}^{f,l}} = f'(a^{f,l} \cdot \xi_{\mathbf{z}}^{f,l} + b^{f,l}) \cdot a^{f,l} \quad (22)$$

The remaining derivative $\partial \xi_{\mathbf{z}}^{f,l}/\partial y_{\mathbf{z}'}^{f',l-1}$ depends on the type of sub-sampling we have chosen – either by averaging using (14):

$$\frac{\partial \xi_{\mathbf{z}}^{f,l}}{\partial y_{\mathbf{z}'}^{f',l-1}} = \frac{1}{\prod_{d=1}^{N} r_s^{l,d}} \quad (23)$$

or with the maximum value according to (15):

$$\frac{\partial \xi_{\mathbf{z}}^{f,l}}{\partial y_{\mathbf{z}'}^{f',l-1}} = \begin{cases} 1 & \text{if } y_{\mathbf{z}'}^{f',l-1} \text{ is the maximum} \\ 0 & \text{otherwise} \end{cases} \quad (24)$$

We are now able to determine the error gradients of the trainable parameters and we can also propagate the calculation of error gradients to layer $l-1$.

2) The convolutional layer: The error G depends on the outputs $y_{\mathbf{z}}^{f,l}$ of neurons $(\mathbf{z}, f, l)$ of the convolutional layer l. Let us first define an auxiliary set S_l':

$$S_l' = \prod_{d=1}^{N} \{0, \ldots, m_d^l - 1\} \quad (25)$$

The derivatives $\partial G/\partial w_{\triangle \mathbf{z}}^{f,f',l}$ can be obtained using the chain rule and (11):

$$\frac{\partial G}{\partial w_{\triangle \mathbf{z}}^{f,f',l}} = \sum_{\mathbf{z} \in S_l'} \frac{\partial G}{\partial y_{\mathbf{z}}^{f,l}} \cdot y_{\mathbf{z}+\triangle \mathbf{z}}^{f',l-1} \quad (26)$$

Note that the derivative $\partial G/\partial y_{\mathbf{z}}^{f,l}$ is assumed to be known.

Denote $OUT(\mathbf{z}, f', l-1)$ the set of neurons from the layer l which are connected to the output $y_{\mathbf{z}}^{f',l-1}$. Let us refer to neurons from this set as $\mathbf{k} \in OUT(\mathbf{z}, f', l-1)$ and to their outputs as $y_{\mathbf{k}}$. Also denote $w_{\mathbf{k}}$ the weight which connects the neuron $(\mathbf{z}, f', l-1)$ to the neuron $\mathbf{k}$. Note that all the neurons indexed by $\mathbf{k}$ are from the layer l. Further, their position in the feature map f in the layer l has to be considered, unlike the simpler record described by the index $\mathbf{z}$.

We will determine the partial derivatives $\partial G/\partial y_{\mathbf{z}}^{f',l-1}$ again by applying the chain rule and (11):

$$\frac{\partial G}{\partial y_{\mathbf{z}}^{f',l-1}} = \sum_{\mathbf{k} \in OUT(\mathbf{z}, f', l-1)} \frac{\partial G}{\partial y_{\mathbf{k}}} \cdot w_{\mathbf{k}} \quad (27)$$

The derivative $\partial G/\partial y_{\mathbf{k}}$ is assumed to be known, as $y_{\mathbf{k}}$ is in fact some of the $y_{\mathbf{z}}^{f,l}$. Herewith, we have derived the terms both for the error gradient and for the partial derivatives in layer $l-1$.

D. Time complexity

The time complexity of training a convolutional and a sub-sampling layer in ND-CNN model will be analyzed.

1) The convolutional layer: The convolutional layer l requires $O(|F_l| \cdot |F_{l-1}| \cdot (r_c^l)^N \cdot \prod_{d=1}^{N} m_d^l)$ computational steps to process its input. This follows from (11) and from the fact that there are $|F_l| \cdot \prod_{d=1}^{N} m_d^l$ neurons in the layer l.

The time complexity to adjust a two-dimensional convolutional layer l for one pattern is $O(|F_l| \cdot |F_{l-1}| \cdot (r_c^l)^2 \cdot m_{l-1} \cdot m_l)$. Now the receptive field has N dimensions of size r_c^l and the size of the feature maps will be $\prod_{d=1}^{N} m_d^l$ (respectively $\prod_{d=1}^{N} m_d^{l-1}$ for the layer $l-1$). Combining these terms together yields the complexity $O(|F_l| \cdot |F_{l-1}| \cdot (r_c^l)^N \cdot \prod_{d=1}^{N} m_d^{l-1})$ for the adjustment of the convolutional layer l in the ND-CNN model for one pattern. Note that the following holds: $\prod_{d=1}^{N} m_d^l \leq \prod_{d=1}^{N} m_d^{l-1}$.

2) The sub-sampling layer: The number of steps required by a sub-sampling layer l to process the input is linear in the number of neurons from the layer $l-1$. Therefore the time complexity of the processing is $O(|F_{l-1}| \cdot \prod_{d=1}^{N} m_d^{l-1})$. A sub-sampling layer in a two-dimensional CNN with feature maps of the size $m_l \times n_l$ required $O(|F_{l-1}| \cdot m_{l-1} \cdot n_{l-1})$ steps to adjust for one input pattern. Analogically, in an N-dimensional case it is $O(|F_{l-1}| \cdot \prod_{d=1}^{N} m_d^{l-1})$ computational steps.

3) Comparison with CNN: In ND-CNN, the size of a receptive field has a greater impact on time complexity than in classical CNN. Nevertheless, in ND-CNN the size of the feature maps shrinks faster and with a lesser number of convolutional and sub-sampling layers than in CNN. The total number of neurons in the network is thus smaller in ND-CNN. Let us assume that both the CNN and ND-CNN have M input neurons corresponding to the total number of pixels in the input image. CNN thus receives input patterns of size $\sqrt{M} \times \sqrt{M}$. We omit the effect of convolutional layers and assume that all sub-sampling layers l have the same sub-sampling parameters $r_x^l = r_y^l = r$. Further, no overlapping of the sub-sampled areas is allowed to occur. Then there must be $\log_r \sqrt{M} = \frac{1}{2} \log_r M$ sub-sampling layers to obtain feature maps with one neuron. For ND-CNN the initial feature map size would be $\sqrt[N]{M}$ and there would be $\frac{1}{N} \log_r M$ layers.

For CNN, let $m_l \times n_l$ denote the size of the feature maps and $m_1'^l \times m_2'^l \times \ldots \times m_N'^l$ in the case of ND-CNN. As the size of the input layer has been assumed to comprise M neurons, we can further conclude that $\prod_{d=1}^{N} m_d'^l \leq m_l \cdot n_l$ for l and N big enough and for receptive fields r_c^l retaining

TABLE II

TIME COMPLEXITY FOR THE l-TH LAYER IN THE CNN AND ND-CNN MODEL: F_l DENOTES THE SET OF FEATURE MAPS IN THE LAYER l, $m_l \cdot n_l$ CORRESPOND TO THE SIZE OF THE FEATURE MAPS IN THE LAYER l OF TRADITIONAL CNN. IN ND-CNN, m_d^l STANDS FOR THE SIZE OF THE FEATURE MAPS FROM THE LAYER l IN THE DIMENSION d. r_c^l DETERMINES THE SIZE OF THE RECEPTIVE FIELDS IN THE LAYER l.

Layer l		***CNN model***	***N*D-CNN model**
Sub-sampling	Weight adjustment	$O\left(\mid F_{l-1} \mid \cdot m_{l-1} \cdot n_{l-1}\right)$	$O\left(\mid F_{l-1} \mid \cdot \prod_{d=1}^{N} m_d^{l-1}\right)$
	Input processing	$O\left(\mid F_{l-1} \mid \cdot m_{l-1} \cdot n_{l-1}\right)$	$O\left(\mid F_{l-1} \mid \cdot \prod_{d=1}^{N} m_d^{l-1}\right)$
Convolutional	Weight adjustment	$O\left(\mid F_{l-1} \mid \cdot \mid F_l \mid \cdot \left(r_c^l\right)^2 \cdot m_{l-1} \cdot n_{l-1}\right)$	$O\left(\mid F_{l-1} \mid \cdot \mid F_l \mid \cdot \left(r_c^l\right)^N \cdot \prod_{d=1}^{N} m_d^{l-1}\right)$
	Input processing	$O\left(\mid F_{l-1} \mid \cdot \mid F_l \mid \cdot \left(r_c^l\right)^2 \cdot m_l \cdot n_l\right)$	$O\left(\mid F_{l-1} \mid \cdot \mid F_l \mid \cdot \left(r_c^l\right)^N \cdot \prod_{d=1}^{N} m_d^l\right)$

the same size. Hence the time complexity of the recall and training phase of a sub-sampling layer l is asymptotically equal both for CNN and ND-CNN. Recall and training of a convolutional layer l in ND-CNN is slower by the factor of $(r_c^l)^{N-2}$ when compared to the CNN model. However, r_c^l is usually a small number equal for all layers l of the network, e.g. $3 \leq r_c^l \leq 5$. Under such circumstances, the difference in the factor $(r_c^l)^{N-2}$ becomes less important. The ND-CNN would actually calculate its output and adjust its weights faster than the CNN.

IV. SUPPORTING EXPERIMENTS

The analyzed 3-dimensional face scans are characterized by plentiful data almost incapable of full processing within a reasonable amount of time. When evaluating the performance of the developed N-dimensional convolutional neural networks, we were thus in particular interested in answering the following questions:

1) To what extent do pre-processing transformations impact the precision of the final classifier?
2) How much does the used architecture (and its complexity) influence the performance of the classifier?
3) Do the ND-CNN networks enhanced with an enforced internal representation support an easier interpretation of the extracted knowledge?

A. Face model data

The source data used in our experiments comprised 122 scanned 3D-face models of real people. The face models were saved in standard VRML format. Each model contained several basic environment definitions (e.g., lighting or texture) and a 3D triangular mesh describing the surface model of the respective face in the space, see Fig. 4.

The positions of the triangles vertices, their normal vectors and the texture mapping represent enough information to display the face models. On the other hand, the scanned face models consist of big amount of tiny triangles. For this

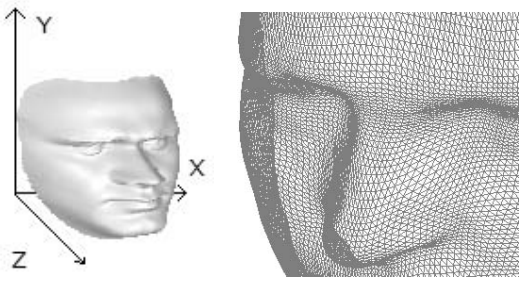

Fig. 4. Face model and its default orientation (left). The face mesh (right) consists of triangles spanned over its vertices.

reason, we used 3 different types of transformations mapping the input data into third-order real-valued tensors during pre-processing. In this way, all the face models were transformed into tensors of a predefined size $22 \times 22 \times 22$ with values from the range $< 0, 1 >$. We will call the components of these tensors voxels (because of their analogy to 2D-image pixels). The value of the voxel will be regarded as its color. The tested transformations comprised:

- Direct transformation provides a natural 3D representation of the processed objects. Lighter grey shades are assigned to the forehead, chin and to the tip of the nose.
- Clustered transformation resembles the direct transformation, except for an additional clustering of voxels. All voxels from the same cluster will be assigned the same color corresponding to their order on the Z-axis.
- SOM-transformation shall pre-extract significant features from the processed models. A similar coloring procedure is used like in the previous case, considering, however, also the size of the formed clusters. Experimental results from Table III confirm more advanced pre-processing for this type of transformation.

To provide enough data for training, testing and validation, the set of tensors was further extended by rotations and resizing. The face models were thus used in their normalized

form (Fig. 5) and also in its re-scaled form, where the original proportions were preserved. Both face models were further set into 5 different positions by rotation around the center point – see Fig. 6. This yields altogether $2 \cdot 5 = 10$ tensors for each original face model. After preprocessing, each input pattern has 22^3 voxels which keeps the computational costs within reasonable limits.

All the tests were performed by means of the FaceDisplay software [12] developed for 3D data processing (analysis and classification of 3D face scans) that is accessible online at https://sourceforge.net/projects/ann3d/. Data pre-processing has been run on personal notebook HP Compaq 6710s, with a 4GB RAM and 32-bit processor Intel Core 2 Duo T7200. A personal computer with AMD Athlon(tm) 64 X2 Dual Core Processor 3800+ and 2GiB of RAM was used for the final computationally intensive evaluations.

B. Evaluation of the ND-CNN model

In ND-CNNs, the first layer stands for the input. In our case, it has one feature map of size $22 \times 22 \times 22$. Further, the network contains three pairs of convolutional and sub-sampling layers, followed by a feed-forward layer and an output layer. Only the first and last layers have a fixed number of neurons (there are two neurons in the output layer, one for each gender). However, the number of feature maps in the respective layers and the number of hidden neurons in a feed-forward layer often need to be specified by trial and error. Training of one ND-CNN model required with the above-specified hardware usually between 5 to 10 hours. For this reason, we wanted to investigate the role of the chosen network architecture in the training process.

In the experiments, the pre-processed data was split randomly into a training, validation and testing set – with the ratio 7:1:3. The training set was used to train the network, while the validation set helped to decide at which moment the network was performing best. Afterwards, the resulting network classified the face models from the test set, which was used to assess the actual performance of the model.

The best-performing candidate networks were further tested by a variant of the k-fold cross validation to determine the mean classification error and its standard deviation. The learning rates were set to 0.1 and the internal representation was enforced with the parameter $c_F = 10^{-4}$ for all of the sub-sampling layer. From Table III, we can see that networks equipped with more feature maps in the first convolutional and sub-sampling layers tend to yield better results.

In addition, the enforced internal representation allows for a transparent network structure. Fig. 8 shows the output of the second feature map from the second sub-sampling layer for two different inputs – for a male model and for a female model. Active neurons indicated by bigger and darker cubes clearly show stronger jaws and arches of the eyebrows with a sharper nose for the man on the left. The face model of the woman on the right is characterized by visibly more gentle features, a smaller rounder nose and a straighter forehead. The most active neurons thus reveal an interpretation that clearly corresponds with the human comprehension of a face.

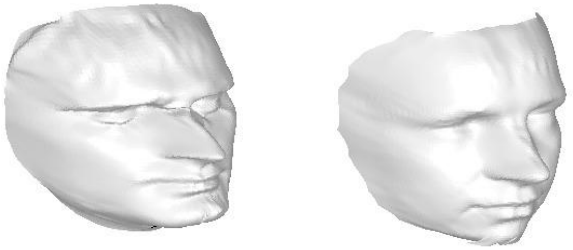

Fig. 5. Normalized face models of a man (left) and a woman (right).

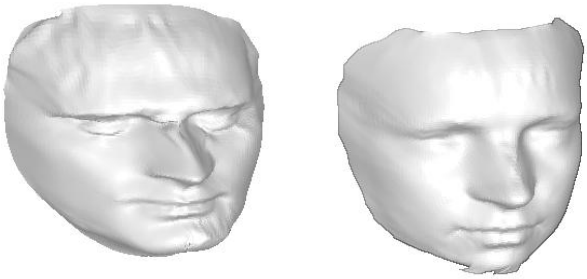

Fig. 6. Rotated face models of a man (left) and a woman (right).

V. Conclusions

A rapid growth of publicly available multimedia data attracts a lot of contemporary research also towards the development of efficient processing strategies applicable to 3D-data. Immense 3D data defining real-world objects are, however, often too big to be analyzed by commonly used classical techniques that are also prone even to small changes in inputs caused, e.g., by rotation or noise. Our goal was thus to propose a new model of convolutional neural networks capable of processing general high-dimensional object data, while also allowing for an adequate interpretation of the extracted knowledge. In particular the latter characteristic could be used later on for semantics assignment when analyzing significant features detected in the data.

The introduced ND-CNN model has been tested on set of 3D face models. In this experiment, our task was to

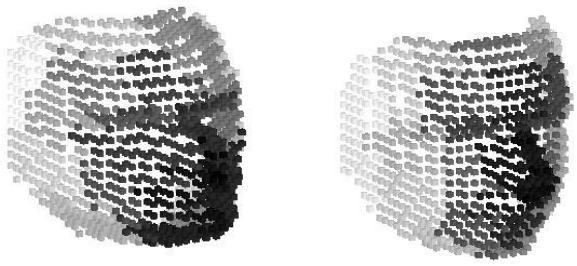

Fig. 7. Normalized face models of a man (left) and a woman (right) after a 3-dimensional SOM-network had been applied for pre-processing. Each of the cubes in the picture corresponds to a neuron; the size of the cube corresponds to a value y of such neuron, while the neurons with zero y value are not displayed.

TABLE III

PERFORMANCE OF THE EVALUATED CANDIDATE ARCHITECTURES FOR GENDER CLASSIFICATION OF 3D-FACE MODELS.

Architecture	***Direct***		***Clustered***		***SOM***		Average training time, 300 epochs (minutes)
	Mean error (%)	Standard deviation	Mean error (%)	Standard deviation	Mean error (%)	Standard deviation	
6, 64, 64 - 24	8.20	1.81	**5.37**	**1.53**	5.25	1.91	581.24
8, 32, 64 - 32	9.02	2.24	5.70	2.16	1.32	0.82	388.65
12, 24, 64 - 24	**8.16**	**1.63**	5.41	1.33	**1.28**	**0.47**	387.68

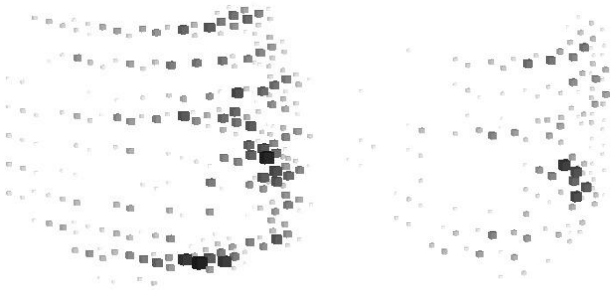

Fig. 8. The output of the second feature map from the second layer of the ND-CNN applied to normalized face model of a man (left) and a woman (right). Active neurons are indicated by bigger and darker cubes.

determine the gender of the 3D face model, which seems to be a challenging task for humans. The obtained results, however, show that the developed ND-CNN model is able to far outperform humans lacking any expertise in the field of anthropology. From the theoretical point of view, the main contribution of this work consists in:

- introducing a new model of ND-CNNs capable of efficient processing of high-dimensional object data without any advanced pre-processing (the original CNNs are tailored to target in particular 2D image data),
- enhancing the discussed ND-CNN model with a strong interpretation power, that allows to assign semantics to significant object parts detected in the analyzed data,
- keeping the computational complexity of ND-CNN at a level comparable to the traditional CNN-model when considering input data of a similar size.

A. Further research

A significant drawback of the general ND-CNN consists in their architecture fixed already before training – both the size of their feature maps and the number of network layers must be defined beforehand. Anyway, finding a convenient configuration for ND-CNN might be a tedious task that requires the user to perform many trials. For this reason, techniques adjusting automatically the network topology [9] might be of advantage in such a case. Within the framework of our further research, we also plan a more extensive evaluation of the proposed technique on considerably larger data sets. Yet other experiments should involve testing of ND-CNNs also for higher-dimensional object data, eg., [7].

ACKNOWLEDGEMENT

This research was partially supported by the Czech Science Foundation under Grant-No. P103/10/0783 and Grant-No. P202/10/1333. Further, the authors would like to acknowledge the support received from the Department of Anthropology and Human Genetics, Faculty of Science, Charles University Prague, that kindly provided the 3D-facial scans used in the experiments.

REFERENCES

[1] D. Ciresan, U. Meier, J. Masci and J. Schmidhuber, "A Committee of Neural Networks for Traffic Sign Competition," *Proceedings of IJCNN 2011*, pp. 1918–1921, 2011.

[2] S.R. Cohen, et al., "Log-linear allometry of fetal craniofacial growth in Down's syndrome," *Journal of Craniofacial Surgery*, vol. 6, pp. 184-189, 1995.

[3] DeepLearning, "Convolutional Neural Networks (LeNet) DeepLearning v0.1 documentation," *online accessed on April, 16, 2012* at: "http://deeplearning.net/tutorial/lenet.html#lenet," 2012.

[4] M. Evison, I. Dryden, N. Fieller, X. Mallett, L. Morecroft, D. Schofield and R.V. Bruegge, "Key parameters of face shape variation in 3D in a large sample," *Journal of Forensic Science*, vol. 55, no. 1, pp. 159-162, 2010.

[5] D.H. Hubel and T.N. Wiesel, "Receptive fields and functional architecture of monkey striate cortex," *The Journal of Physiology*, vol. 195, no. 1, pp. 215-243, 1968.

[6] S. Ji, W. Xu, M. Yang and K. Yu, "3D Convolutional Neural Networks for Human Action Recognition," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 35, no. 1, pp. 221-231, 2013.

[7] A. Kurjak, B. Miskovic, W. Andonotopo, M. Stanojevic, G. Azumendi and H. Vrcic, "How useful is 3D and 4D ultrasound in perinatal medicine?" *J. of Perinatal Medicine*, vol. 35, no. 1, pp. 10-27, 2007.

[8] 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.

[9] I. Mrazova and M. Kukacka, "Can Deep Neural Networks Discover Meaningful Pattern Features?" *Complex Adaptive Systems*, vol. 2, no. 12, pp. 194-199, 2012.

[10] I. Mrazova and Z. Reitermanova, "Enforced knowledge extraction with BP-networks," *Intelligent Engineering Systems through Artificial Neural Networks*, vol. 17, ASME Press, New York, pp. 285-290, 2007.

[11] I. Mrazova and D. Wang, "Improved generalization of neural classifiers with enforced internal representation," *Neurocomputing*, vol. 70, no. 16-18, pp. 2940-2952, 2007.

[12] J. Pihera, "FaceDisplay," *online accessible* at: "https://sourceforge.net/projects/ann3d/," 2012.

[13] P. Sermanet and Y. LeCun, "Traffic Sign Recognition with Multi-Scale Convolutional Networks," *Proc. IJCNN 2011*, pp. 2809–2813, 2011.

[14] J. Veleminska, et al, "Surface facial modelling and allometry in relation to sexual dimorphism," *Homo*, vol 63, no. 2, pp. 81-93, 2012.

[15] T.L. Wilson, K. Rohlfs and S. Huttemeister, "Tools of radio astronomy," Springer, 2009.