in: Proc. Workshop Farbbildverarbeitung (FWS), Ilmenau, Germany, pp. 169-182, 2016

# Are Color Features Important for Person Detection? – Insights into Features Learned by Deep Convolutional Neural Networks

Markus Eisenbach, Daniel Seichter, Horst-Michael Groß

Ilmenau University of Technology,
Neuroinformatics and Cognitive Robotics Lab,
98684 Ilmenau, Germany
markus.eisenbach@tu-ilmenau.de

**Abstract** Robust person detection is required by many computer vision applications. State of the art hand-crafted features rely on texture only or make only limited use of color. We deliver insights into features extracted by a deep learning approach, that combines three Convolutional Neural Networks to detect people at different scales. The networks learn features from raw pixel information. Thus, they do not rely on hand-crafted features, but decide by their own, which features are important. Since the networks combine color and texture features, the color information seems to be valuable. We observed, that a specific color space was constructed, that can improve person detection. We present this color space and show how the deep neural networks make use of it.

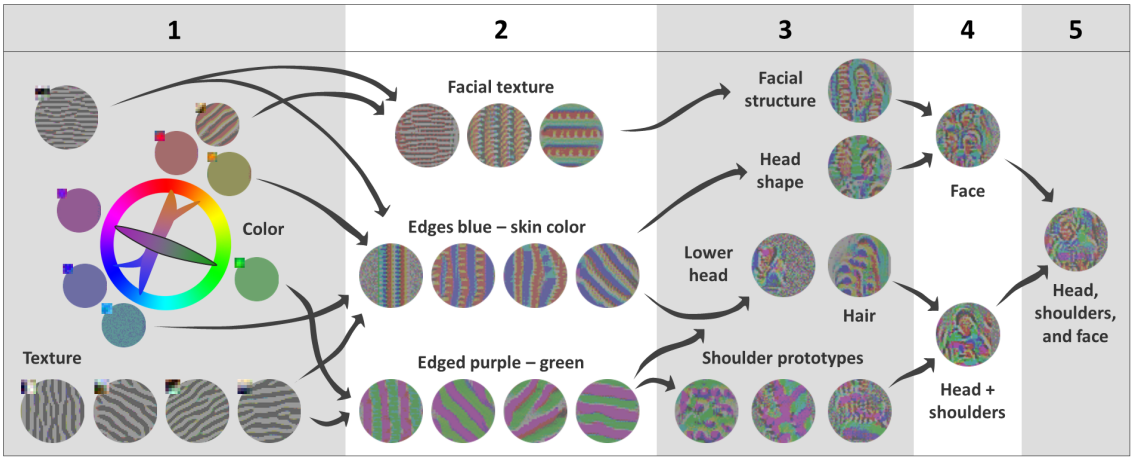



Figure 1: Combination of learned texture and color features to detect persons.

# 1 Introduction

Detecting persons in images at high accuracy is indispensable for a wide range of applications, including pedestrian detection for car assistance systems [5], person detection for automatic surveillance video analysis [14], and potential user recognition for human robot interaction [21]. Therefore, a person detector should be generic and applicable to several domains. In the last years, a lot of approaches have been presented using different advanced hand-crafted features. Recently, deep learning approaches supplant these methods by learning superior features data-driven. Following this trend, in [6] we presented an approach that was trained on a large dataset including samples from several domains. It outperforms the state of the art (see Fig. 2 for

Figure 2: Person detections by Convolutional Neural Networks at multiple scales shown as red, blue, and green boxes (for near, medium and far persons) with overlaid network output. Examples are taken from the Caltech dataset [5]. The overlaid network output is coded as follows: Red regions represent a high output for person class while yellow/green/blue regions represent a low output in this scale. The three cooperative Convolutional Neural Networks perform very well in detecting nearly all persons at different scales in the scene while false positive detections are not present. For a extensive evaluation and comparison to state of the art approaches, we refer to [6].

exemplary visual results), including approaches that used a large variety of advanced hand-crafted features. In contrast, our approach operates on raw pixels and learns the feature representation by its own. Therefore, we assume, that the better performance is a result of the superior features. In this paper, we will analyze the learned features of this deep learning approach, to find out, whether color plays an important role. This would call the state of the art hand-crafted features that only rely on texture into question. Furthermore, we try to analyze, how these hand-crafted features could be improved by the use of color.

# 2 Analyzed Deep Learning Approach

To analyze, whether color is important for person detection, we will have a closer look at our deep learning approach presented in [6]. It detects persons at multiple scales using three Convolutional Neural Networks (CNNs). In the following, a short review of the basic concepts is presented.

## 2.1 System Overview

For detecting persons at different scales, we decided in favor of a hybrid approach, known as multi-resolution model, that uses a resolution pyramid in combination with detectors at different scales. Thus, it is fast in the application phase, and the number of neural networks to be trained is manageable.
Our approach uses three Convolutional Neural Networks. We trained these networks on cropped images showing persons (positive class), other objects, and typical false detections (e.g. sub-images containing only parts of a person or persons that fill only parts of the image section = negative class). After network training, fully connected layers were converted to convolutional layers to be able to process images of any size without the need to shift a sliding window to several locations. Thus, we achieve a

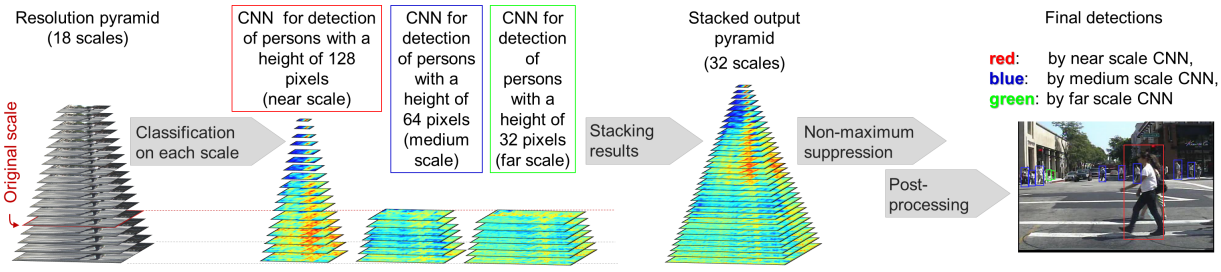



Figure 3: Processing chain in the application phase. To create the resolution pyramid, the input image is scaled such that the image size is exactly halved after seven scales. Thus, each of the three CNNs has to process seven scales. The near scale CNN (red box, second from left) additionally processes smaller scales to detect larger persons. The classification results are stacked and non-maximum suppression is applied to find the best fitting positions and scales for all persons in the scene. Finally, detections are post-processed to remove some false positives.

great speedup in the application phase.
Fig. 3 shows the processing chain of the application phase. Each of the Convolutional Neural Networks processes multiple scales of the resolution pyramid. For each image scale, an output map is calculated. High neural activations suggest that persons are present in that region of the image (see Fig. 2). When these classifications have been done, the full output pyramid can be constructed. Then, a 3D non-maximum suppression (NMS) is applied to find persons in the scene and at the best fitting scale. For NMS, we implemented an approximation of the mean-shift algorithm as a single 3D pooling layer. In a last step, we post-process all detections. Therefore, we filter out detections that do not fit the ground plane and those ones, that do appear only once in consecutive frames.
For implementation, we used the Theano framework [1, 2] in combination with Keras[1]. The hardware, used for training, is a PC with a Core i7 CPU, 16 GB RAM, and a single NVIDIA Titan X GPU.

## 2.2 Network Architecture and Processing Chain

To apply the scheme presented above, each of the three CNNs processes parts of the resolution pyramid, as shown in Fig. 3.
The objective of the network design is to get network outputs for different scales that are easily comparable such that a 3D NMS is sufficient to locate persons in the input image. The outputs would be comparable if a single network would produce all output maps. But a single network would not be flexible enough to detect persons at all scales adequately. Therefore, we decided in favor of a collection with three CNNs to detect persons at near, medium, and far scale, each producing parts of the output pyramid. The three CNNs process input image patches of different size, so each one specializes on detecting persons of a specific size. In order to get output maps that appear to be produced by the same network a proper network architecture for each of the networks must be chosen. However, the network design is not the focus of this paper. Thus, we refer to [6].

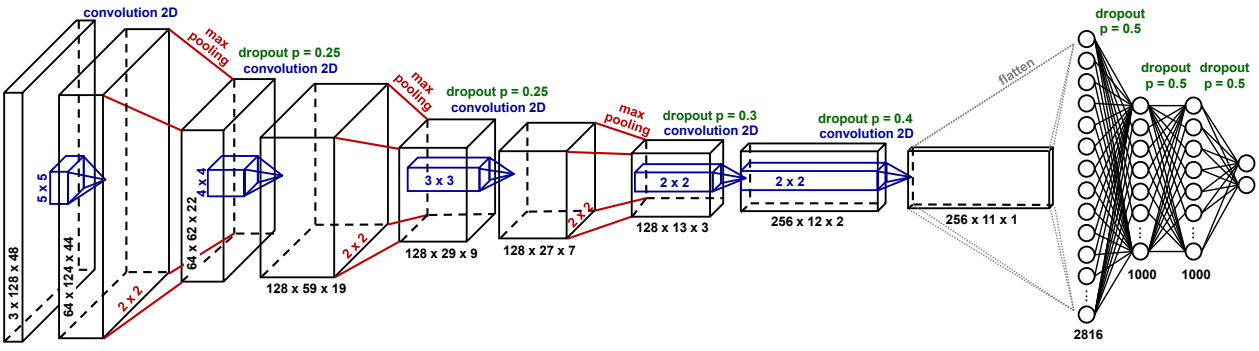



Figure 4: Convolutional Neural Network topology for detecting persons at near scale. The network is responsible for detecting persons of height 80 pixels and larger. The input is a RGB image of size $128 \times 48$ pixels.

[1]Keras: Deep Learning library for Theano and TensorFlow `http://keras.io/`

In this paper, we only analyze the features of the near scale CNN. It processes the largest input images. Therefore, its features are best to visualize. For the network topology, see Fig. 4.
The near scale CNN (input height 128 pixels) has five convolutional layer and three pooling layers. The main objective of these layers is the feature extraction. Additionally, the network has two fully connected layers and a softmax output layer consisting of two neurons representing non-persons and persons. These layers build on the learned features and accomplish the classification task. All neurons use scalar product activation and a rectified linear output function (ReLU).

## 2.3 Input Coding

As input we take the pixels of the image to be classified in RGB color space. We normalize the input such that a black pixel is represented by three neurons (RGB) with activation -1, white ones by three neurons with activation 1, and medium gray pixels by three neurons with activation 0. Thus, the latter have no influence to subsequent layers. We decided in favor of this input coding to accomplish that normalization has zero mean. By gray world assumption the expected mean color is gray.
Additionally, we use zero padding for training samples that do not fill the complete patch (e.g. when persons are near the camera and legs are outside the image). Thus, the added regions do not have any influence on subsequent layers.

## 2.4 Network Training

For training the Convolutional Neural Networks, we used cropped images of size $128 \times 48$ showing both persons and non-persons. To get a large, versatile, general purpose training dataset, we combined 22 datasets from pedestrian detection and person re-identification domain (see [6] for a listing).
Negative samples were taken from the INRIA [3], NICTA dataset [15] and from publicly available images without persons showing landscapes and urban scenes. To simulate typical false detections, we extracted misaligned patches of the SAIVT-SoftBio dataset [12] (see [6] for details on patch extraction). To also collect real false detections that are made by state of the art detectors (we used [4, 9] and [19]), we recorded data with a mobile robot. The autonomous robot drove through a clinic [10, 11] and a faculty building [18] when no people were present, so every detection represents a false detection. Then, the robot drove through the corridor of the clinic when lots of persons were present. A laser-based detection approach [20] and the map of the scene were used to check the detections for plausibility [7]. We additionally checked them manually. These negative samples helped the networks to avoid typical errors.
Summarized, we collected a relatively large training dataset containing 100,107 positive and 628,636 negative samples, which was crucial to learn proper features and classifiers using a deep learning approach.
As training algorithm, we used stochastic gradient descend (SGD) with mini-batches and momentum. To avoid overfitting, we used dropout for regularization [13, 17]. It was applied to all layers, except the input layer, using relatively large dropout rates. See [6] for details.

### 2.5 Reshaping CNNs for Application Phase

After network training, we reshaped the CNNs such that they are applicable to different image sizes. Therefore, the weights of the fully connected layers were reshaped to three dimensions such that they can be used as filter kernels. This avoids the need of shifting the sliding window to several positions and, thus, speeds up the processing in the application phase. Details can be found in [6].

### 2.6 Non-Maximum Suppression and Post-processing

When CNNs are reshaped, they can be used directly to calculate output maps from differently scaled full images instead of patches. Then, the outputs of the three CNNs can be stacked to create the output pyramid. Finally, non-maximum suppression has to be applied to find the best positions and scales for all persons in the scene. Therefore, we implemented a single 3D max-pooling layer as approximation of the mean-shift algorithm.
If a maximum is found, a person is detected and the scale and position in the output volume can be used to localize its position. The position in the original image is computed as the output neurons' receptive field in the input layer, which matches the detected person's bounding box.
As post-processing steps, we filter out detections that do not fit the ground plane and those ones that do appear only once in consecutive frames.
Again, for details, we refer to [6].

## 3 Feature Visualization Methodology

To visualize the features of the CNN, we have implemented two methods. First, we visualized the low level features directly by plotting the neurons' weights. Second, we followed the idea of [8] to use gradient ascend to modify a randomly initialized input image such that the output of a neuron is maximized. This idea has also been addressed in recent work, e.g. [16]. Also the popular deep learning visualization toolbox [22] includes this technique.
The first method is straightforward. The weights of the first convolutional layer are connected to the input. Therefore, they can be interpreted as features directly.
All features of consecutive layers are a (non-linear) function of features from previous layers. Therefore, they cannot be visualized directly. Instead, we try to generate inputs, that maximize the output of selected neurons. If the neuron's output is high, the input image shows, what the neuron is looking for. Therefore, the feature represented by the neuron can be guessed from that image.
To generate such an image, we start with random normally distributed noise near a gray image, which represents the zero (non relevant) input. Then, the image is presented to the neural network. During this forward pass, the activation of the neurons in each channel is calculated. Then, in a second step the gradient with respect to the input image is used to adapt the input in order to maximize the activation of

the neurons[2]. This procedure is very similar to [8].
We recognized, that the activation maximization (AM) images for all output channels in the first convolutional layer could be created, but the approach failed for most of the subsequent layers due to very complex features. The reason is, that a randomly generated input image does not activate very specialized neurons at all. Therefore, we analyzed the weights connecting the neurons of two layers and created the initial input image from AM images from the previous layer. We identified the neurons in the previous layer that can have the most influence on the analyzed filter response, considering the weights between these layers and the maximized activation of the neurons in one channel in the previous layer. Then, we constructed the initial image by randomly sampling from the AM images of these identified neurons. After this adaption, we were able to visualize the features for most of the neurons in our CNN, including very advanced high level features.

# 4 Feature Analysis

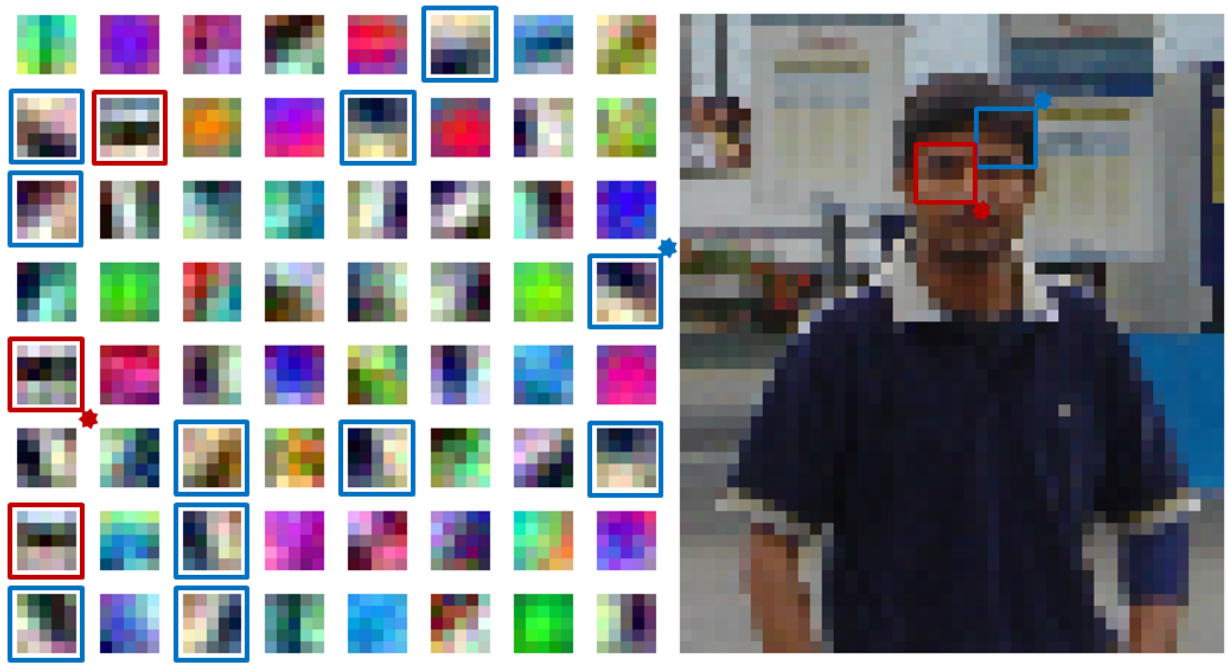


Figure 5: Filters of the first convolutional layer learned by the near scale CNN. Left: The 64 learned filters. Right: Sample image from the INRIA dataset to set the filter size in relation to the patch size. Red and blue boxes highlight specialized filters. For two filters good fits in the sample are show.

Fig. 5 shows the 64 filters of the first convolutional layer learned by the near scale CNN. The images that maximize the activation of the filters are shown in Fig. 6. The network learned typical color filters and textural filters, as well as some texture filters in different color channels. It is remarkable, that it additionally learned some specialized filters. Blue boxes highlight filters that search for skin color and edges of elliptical structure simultaneously. These can be combined to find the contour of a

[2]Actually, the mean activation of all neurons in one channel is maximized.

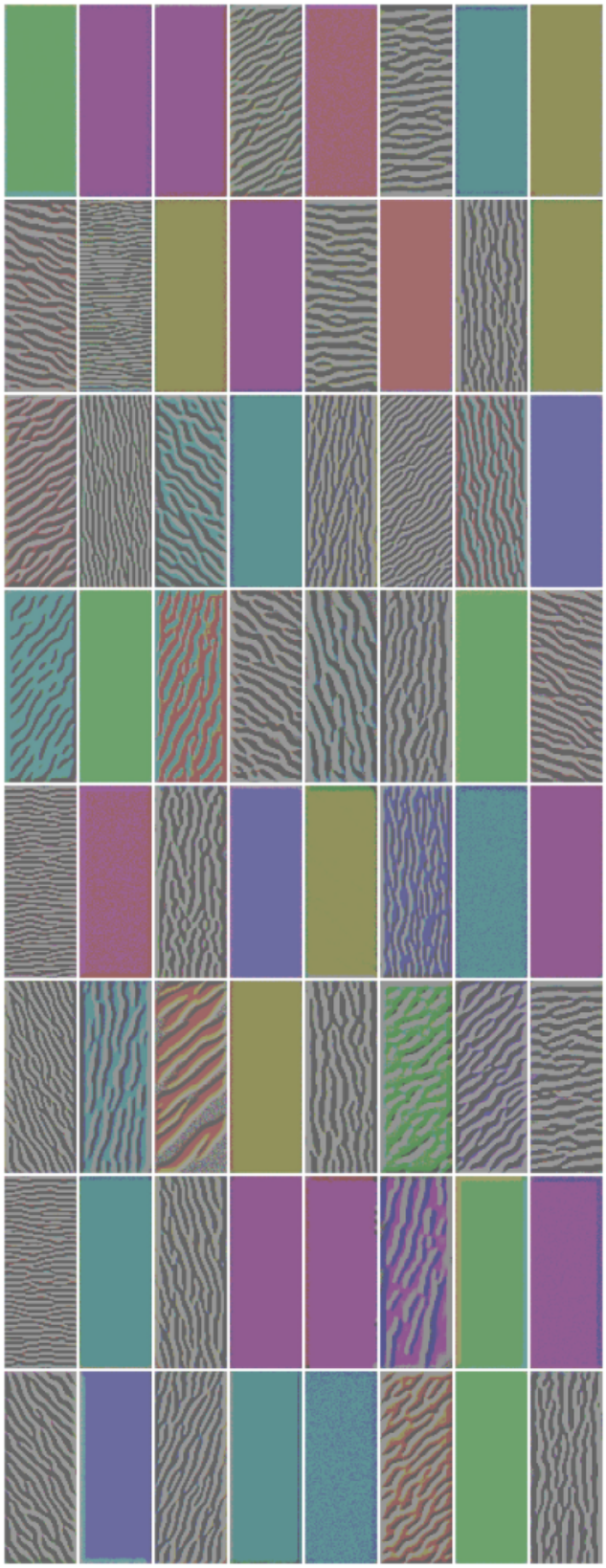
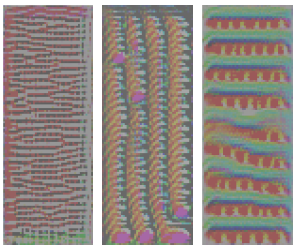
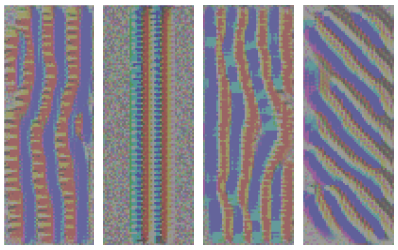


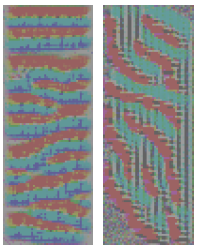
Figure 6: Artificially generated input images for maximizing the response of the 64 learned filters in the **first convolutional layer**. It can be seen, that three types of features have been learned: Texture features in different orientations and frequencies, pure color filters, and a mixtures of both, which are texture features in a specific color space.



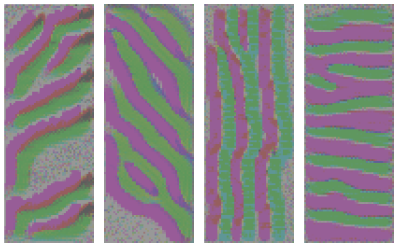
(a) Facial texture



(b) Skin color – blue edges



(c) Red – cyan edges

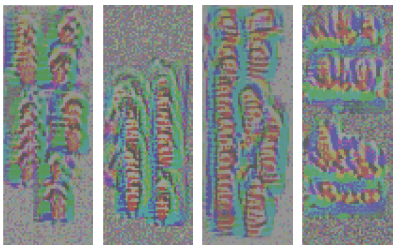


(d) Purple – green edges

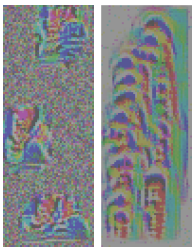
Figure 7: Artificially generated input images for maximizing the response of the learned filters in the **second convolutional layer**. Exemplarily, 13 of 128 filters are shown, that represent typical characteristics. (a) Features searching for facial structure. These are combined with structural filters to find faces in subsequent layers. (b) Edge filters to find the contour of the face. These features search for high frequent horizontal texture and edges of defined orientation in a lower frequency. The edge must be in the color axis skin color – blue. (c) Red – cyan edges are also used to find the contour of the face. (d) The majority of features are edges in purple – green color axis. These are used to find edges in all orientations in different frequencies, most often horizontal orientation in low frequencies. They are combined to find the shoulders in subsequent layers. Fig. 1 shows how these features are combined. Additionally, some features of the first layer are taken to the second layer without changes. Therefore, also some pure texture features are present.

face, as can be seen in Fig. 1. Red boxes highlight filters that search for defined lines. These are used to find facial structure in the second layer (see Fig. 7). Note that these filters are very similar to haar-like features learned by the famous Viola & Jones face detector.

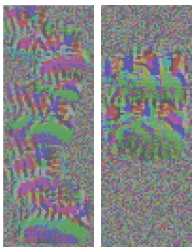
We further examine these specialized low level features by looking at the AM images of the filters in the subsequent layers. Fig. 1 shows how filters are combined, to locate the face of a person. In the first layer, high frequent horizontal texture is extracted to find facial texture. Additionally, skin colors and their inverse are extracted. These are the red, yellow, blue, and cyan color filters marked in the color circle in Fig. 1. The other features in the first layer are used to find texture and shapes in the subsequent layers. To find these shapes, not the gray values are uses, but an orthogonal color axis to the skin color – blue axis. Therefore, purple, magenta and green color filters are learned.



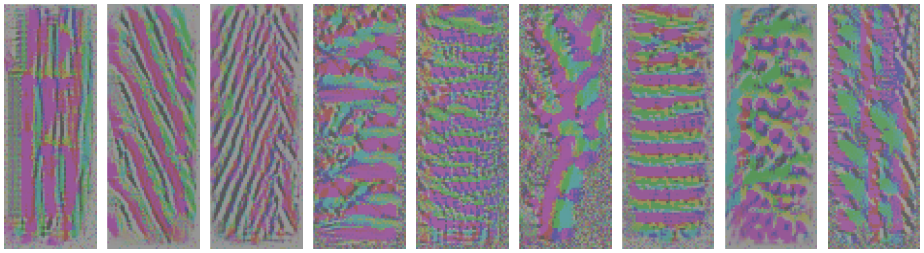
(a) Facial structure



(b) Head



(c) Shoulders

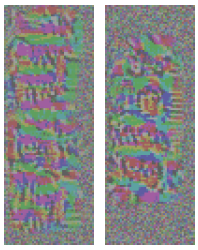


(d) Purple – green shapes

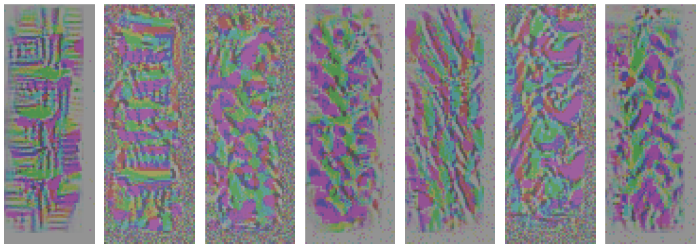
Figure 8: Artificially generated input images for maximizing the response of the learned filters in the **third convolutional layer**. Exemplarily, 17 of 128 filters are shown, that represent typical characteristics. (a) Features searching for facial structure. At this layer the facial texture is inside a raw head shape. Note, that convolutional filters are positionally invariant. (b) Edge filters to find the contour of the face. These features are specialized to find parts of the face and are combined to a complete face in the next layer. The edge must be in the color axis skin color – blue. (c) Similar to face contour filters, these filters search for the shoulders and the neck. The strong shoulder edge uses the purple – green color axis. (d) Again, the majority of features are shapes in purple – green color axis. Edges of the previous layer are combined to find special parts of the full body. Fig. 1 shows how these features are combined to higher level features in the next layer. Note, that the third convolutional layer is the first one, that only contains combined features. This means, that no pure texture or color features are present.

In the second convolutional layer, the color and texture features are combined (see Fig. 7). The third convolutional layer shows the presence of mid level features, that represent raw shapes of heads, shoulders and body parts (see Fig. 8). Additionally, the facial texture is combined with head shapes. In this layer, the blue and skin colors are used exclusively for head shapes and the purple and green colors are associated with body shapes and texture. This supports our hypothesis of a color space with two dominant axes (blue – skin color, purple – green).
In the fourth convolutional layer, the shapes are put together to head and shoulder contours (see Fig. 9). Also either facial texture or shoulders are added to the head.
Finally, the fifth convolutional layer puts all together to a complete upper body shape (see Fig. 10). Also the body shapes in the purple – green axis have similarities with body parts.
It is remarkable, that the CNN has learned these high level feature, although the face

(a) Head with face or shoulders



(b) Shape and head mixed



(c) Advanced purple – green shapes

Figure 9: Artificially generated input images for maximizing the response of the learned filters in the **fourth convolutional layer**. Exemplarily, 15 of 256 filters are shown, that represent typical characteristics. (a) Features searching for the head of a person (at any position). In this layer the full contour is recognized. Additionally, in most filters either the facial structure or the transition to the shoulders is visible. (c) The majority of features searches for body shapes in purple – green color axis. The shapes of the previous layer are combined to even more advanced shapes. (b) In this layer, also some hybrid features are present, that search for faces and horizontal edges at the same time. This may be helpful to find the exact position of the face in the classification layers. Fig. 1 shows which features of previous layers are used to get this high level features.

position was not labeled in the data. This means, the deep network has recognized the face as underlying structure that strongly indicates the presence of a person. Since none of the filters is large enough to detect a face at a whole, the network learned to combine multiple features to master this hidden task of finding faces.
But this link of face structure to the presence of a person has also a drawback. We observed some typical false detections that strongly respond to round (face-like) objects with red shades (skin colors) in the upper third of the patch. In the Caltech road traffic scenario, these are back lights of cars and red traffic lights. These objects caused about one third of all false positive detections. Note that green traffic lights do not lead to false detections. For benchmarking details on Caltech dataset see [6].

# 5 Conclusion

In this paper, we have analyzed the features extracted by a deep learning approach that combines three Convolutional Neural Networks to detect people at different scales. The networks learn features from raw pixel information. Thus, they do not rely on

(a) Head with face and shoulders in various poses (b) Shape features

Figure 10: Artificially generated input images for maximizing the response of the learned filters in the **fifth convolutional layer**. Exemplarily, 8 of 256 filters are shown, that represent typical characteristics. (a) Features capable of locating the upper body of a person (at any position). Many filters are specialized to typical head poses. Others are specialized on groups of people. In this layer all high level features from the previous layer are combined. Therefore, most of the images show head shape in combination with facial structure and the transition to the shoulders. Note, that the head composed of edges in the skin color – blue color axis and the edge of the shoulder is serached for in the purple – green color axis. (b) The majority of the features are specialized to find body parts and characteristic edges. These include combinations of visible skin (e.g. at arms, hands, or the head) with defined edges (e.g. orientation of an arm etc.). Fig. 1 shows the chain of combination from low level texture and color features to the high level features in this layer.

hand-crafted features, but decide by their own, which features are important. We have shown, that the CNNs have used color information in combination with texture filters to detect persons. Therefore, high level features are constructed from low level color and texture features. The neural network has recognized the hidden task to find faces and used this feature to get a superior performance. To accomplish this task, the network learned to construct a meaningful color space. It consists of two axis. One axis spans from skin color to the inverse blue color. The second axis, that is almost orthogonal to the first axis in the HS(V) color space, spans form purple to green. The skin color axis is used in combination with specialized texture features to find facial structure. In contrast, the purple–green axis is mainly used to extract defined edges and thus the silhouette of a person. This finding contradicts the common practice of applying texture-based detectors to all channels of a color image or to apply it on a gray value image. Instead, the texture-based detectors should be used on a purple–green channel, and other, more specialized features should be used to find the head and shoulders. Additionally, state of the art hand-crafted features should not only rely on texture, but make use of color information, especially skin and hair color.

# References

[1] Frédéric Bastien, Pascal Lamblin, Razvan Pascanu, James Bergstra, Ian J. Goodfellow, Arnaud Bergeron, Nicolas Bouchard, and Yoshua Bengio. Theano: new

features and speed improvements. Deep Learning and Unsupervised Feature Learning NIPS 2012 Workshop, 2012.

[2] James Bergstra, Olivier Breuleux, Frédéric Bastien, Pascal Lamblin, Razvan Pascanu, Guillaume Desjardins, Joseph Turian, David Warde-Farley, and Yoshua Bengio. Theano: a CPU and GPU math expression compiler. In *Proceedings of the Python for Scientific Computing Conference (SciPy)*, June 2010.

[3] Navneet Dalal and Bill Triggs. Histograms of oriented gradients for human detection. In *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 886–893, 2005.

[4] Piotr Dollar, Serge Belongie, and Pietro Perona. The fastest pedestrian detector in the west. In *British Machine Vision Conf. (BMVC)*, pages 68.1–68.11, 2010.

[5] Piotr Dollar, Christian Wojek, Bernt Schiele, and Pietro Perona. Pedestrian detection: A benchmark. In *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 304–311, 2009.

[6] Markus Eisenbach, Daniel Seichter, Tim Wengefeld, and Horst-Michael Gross. Cooperative multi-scale convolutional neural networks for person detection. In *IEEE World Congress on Computational Intelligence (WCCI)*, pages 267–276, 2016.

[7] Markus Eisenbach, Alexander Vorndran, Sven Sorge, and Horst-Michael Gross. User recognition for guiding and following people with a mobile robot in a clinical environment. In *Int. Conf. on Intelligent Robots and Systems (IROS)*, pages 3600–3607, 2015.

[8] Dumitru Erhan, Yoshua Bengio, Aaron Courville, and Pascal Vincent. Visualizing higher-layer features of a deep network. Technical report, Technical Report 1341, University of Montreal, 2009.

[9] Pedro F Felzenszwalb, Ross B Girshick, David McAllester, and Deva Ramanan. Object detection with discriminatively trained part based models. *IEEE Trans. on Pattern Analysis and Machine Intelligence (TPAMI)*, 32(9):1627–1645, 2010.

[10] Horst-Michael Gross, Andrea Scheidig, Klaus Debes, Erik Einhorn, Markus Eisenbach, Steffen Mueller, Thomas Schmiedel, Thanh Quang Trinh, Christoph Weinrich, Tim Wengefeld, Andreas Bley, and Christian Martin. Roreas: robot coach for walking and orientation training in clinical post-stroke rehabilitation: Prototype implementation and evaluation in field trials. *Autonomous Robots*, pages 1–20, 2016.

[11] Horst-Michael Gross, Andrea Scheidig, Markus Eisenbach, Thanh Quang Trinh, and Tim Wengefeld. Assistenzrobotik für die gesundheitsassistenz - ein beitrag zur evaluierung der praxistauglichkeit am beispiel eines mobilen reha-roboters. In *German AAL Conference (AAL)*, pages 58–67. VDE, 2016.

[12] Michael Halstead, Simon Denman, Sridha Sridharan, and Clinton B. Fookes. Locating people in video from semantic descriptions : A new database and approach. In *Int. Conf. on Pattern Recognition (ICPR)*, pages 24–28, 2014.

[13] Geoffrey E Hinton, Nitish Srivastava, Alex Krizhevsky, Ilya Sutskever, and Ruslan R Salakhutdinov. Improving neural networks by preventing co-adaptation of feature detectors. *arXiv preprint arXiv:1207.0580*, 2012.

[14] Alexander Kolarow, Konrad Schenk, Markus Eisenbach, Michael Dose, Michael Brauckmann, Klaus Debes, and Horst-Michael Gross. APFel: The intelligent video analysis and surveillance system for assisting human operators. In *IEEE Int. Conf. on Advanced Video and Signal-Based Surveillance (AVSS)*, pages 195–201. IEEE, 2013.

[15] Gary Overett, Lars Petersson, Nathan Brewer, Lars Andersson, and Niklas Pettersson. A new pedestrian dataset for supervised learning. In *Intelligent Vehicles Symposium (IV)*, pages 373–378, 2008.

[16] Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. Deep inside convolutional networks: Visualising image classification models and saliency maps. *arXiv preprint arXiv:1312.6034*, 2013.

[17] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958, 2014.

[18] Ronny Stricker, Steffen Mueller, Erik Einhorn, Christof Schroeter, Michael Volkhardt, Klaus Debes, and Horst-Michael Gross. Konrad and Suse, two robots guiding visitors in a university building. In *Autonomous Mobile Systems (AMS)*, pages 49–58, 2012.

[19] Christoph Weinrich, Christian Vollmer, and Horst-Michael Gross. Estimation of human upper body orientation for mobile robotics using an svm decision tree on monocular images. In *Int. Conf. on Intelligent Robots and Systems (IROS)*, pages 2147–2152, 2012.

[20] Christoph Weinrich, Tim Wengefeld, Christof Schroeter, and Horst-Michael Gross. Generic distance-invariant features for detection of people with walking aid in 2d range data. In *Int. Symp. on Robot and Human Interactive Communication (RO-MAN)*, pages 767–773, 2014.

[21] Tim Wengefeld, Markus Eisenbach, Thanh Quang Trinh, and Horst-Michael Gross. May i be your personal coach? bringing together person tracking and visual re-identification on a mobile robot. In *Int. Symposium on Robotics (ISR)*, pages 141–148. VDE, 2016.

[22] Jason Yosinski, Jeff Clune, Anh Nguyen, Thomas Fuchs, and Hod Lipson. Understanding neural networks through deep visualization. *arXiv preprint arXiv:1506.06579*, 2015.