Fang *J Big Data* (2017) 4:40
DOI 10.1186/s40537-017-0101-8

Journal of Big Data

METHODOLOGY Open Access

Understanding deep learning via backtracking and deconvolution

Xing Fang*

*Correspondence:
xfang13@ilstu.edu
School of Information Technology, Illinois State University, Normal, IL, USA

Abstract

Convolutional neural networks are widely adopted for solving problems in image classification. In this work, we aim to gain a better understanding of deep learning through exploring the miss-classified cases in facial and emotion recognitions. Particularly, we propose the backtracking algorithm in order to track down the activated pixels among the last layer of feature maps. We then are able to visualize the facial features that lead to the miss-classifications, by applying the feature tracking algorithm. A comparative analysis of the activated pixels reveals that for the facial recognition, the activations of the common pixels are decisive for the result of classification; for the emotion recognition, the activations of the unique pixels indeed determine the result of classification.

Keywords: Deep learning, Convolutional neural networks, Backtracking, Deconvolution

Introduction

Deep learning is to apply computational models to learn data representations with multiple levels of abstraction [1]. The most common deep learning models consist of two different neural network architectures: convolutional neural networks (ConvNets) have been widely applied in computer vision for learning 2D images [2–6] or 3D objects [7]; since recurrent neural networks (RNNs) are good at capturing temporal features from sequential input data, they are often used as language models or translation systems [8–14].

Despite the success of deep learning, gaining insight into the models' internal operations and their behaviours has become an interesting topic in deep learning. As one of the visualization techniques proposed by Zeiler and Fergus [15], deconvolution enables us to observe which features of data that are learned by a trained model. Indeed, the deconvolution provides top–down projections by mapping activations in feature maps generated in intermediate layers back to the input pixel space, showing the patterns captured by the feature maps.

In this work, we aim to gain a better understanding towards deep learning through interpretations of the miss-classified cases in facial and emotion recognitions. We start with proposing a ConvNet and its deconvolutional counterpart. After the network is

trained, a backtracking algorithm is applied in order to trace the activated pixels residing in the feature maps. Those activated pixels are then projected back to the input pixel space, using the feature tracking algorithm, which enables us to observe the facial features that determine the classification results. For facial recognition, we discover that same facial features can lead to different classification results due to different pixel values that they hold; For emotion recognition, we are able to track down the facial features that lead to the miss-classifications.

The rest of the paper is organized as follows: we discuss about the data collection and the process of convolution and deconvolution in "Methodology" section; The miss-classified cases are analyzed in "Preliminary evaluation" section; A detailed description towards the backtracking process as well as the visualization of the experimental results are covered in "Analysis using backtracking and deconvolution" section; we then make a discussion and conclude our work.

Related work

It is a common practice to use visualizations to gain intuition about features that have been learned by a trained ConvNet. Zeiler and Fergus [15] introduced the process of deconvolution as a technique to visualize features that are learned by a ConvNet. Different from the original approach, where a deconvolutional network is used as a way of performing unsupervised learning [16], the process requires a trained ConvNet. Noh et al. [17] proposed a deconvolutional network for semantic segmentation, which is to partition an image into semantically meaningful parts. The deconvolutional layers no longer rely on the parameters from the trained ConvNet. Instead, both the convolutional and deconvolutional layers are trained together, such that the deconvolutional layers are served as decoders that are solely mapping the convoluted images back to their input pixel space.

Methodology

Data collection and preprocessing

Data used in this work is collected from the Radbound Faces Database (RaFD) [18]. The entire dataset contains facial images of 67 subjects that are in different ages (adults and children), genders (male and female), and races (Caucasian and Moroccan). There are 120 images for a subject, where the images were taken from five camera angles simultaneously. Within those images, the subject was trained to show their emotions in three different gaze directions. The emotions are "anger", "disgust", "fear", "happiness", "sadness", "surprise", "contempt", and "neutral". Figure 1 shows a Caucasian female in five different angles. Figure 2 shows the eight different facial expressions.

We exclude the images that are taken in both 0 and 180 angles, since they contain little facial information. The original images have the size of 681 pixels as the width and 1024 pixels as the height. Extra information such as hair and torso, is also included in the images. In order to let the model focus on learning facial information, only faces are extracted from the original images. The facial images are then cropped into colour (RGB) pictures that have the size of 224×224. For the facial recognition, the training set is formed by randomly selecting 12 images per angle per subject; For the emotion

Fig. 1 **a** 180°, **b** 135°, **c** 90°, **d** 45°, **e** 0°

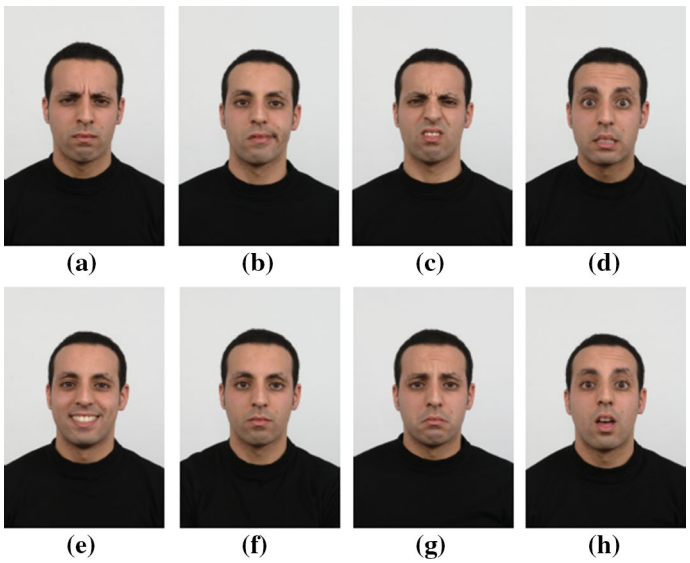

Fig. 2 Eight different emotions: **a** angry, **b** contemptuous, **c** disgusted, **d** fearful, **e** happy, **f** neutral, **g** sad, **h** surprised

recognition, 101 images are randomly selected from each type of emotion under a particular angle. We respectively use the rest of the images as the testing set.

The convolutional network

The architecture of our ConvNet is shown in Table 1. The network takes a fixed-size 224 × 224 RGB image as its input. The image is then passed through a sub-structure that consists of a stack of two convolutional layers (conv) and a max-pooling layer. Each of the convolutional layer has the size of the receptive field set to 3 × 3 with the padding is set to 1 and the non-linear activation function is chosen to be as the rectified linear unit (ReLU). 32 feature maps are generated after the convolution. The max-pooling layer performs over a 2 × 2 filter with stride 2. A stack of three sub-structures is followed by four fully-connected (FC) layers, where the first three layers respectively have 1024, 512, and 256 neurons with ReLU as the activation. The last FC layer has the same number of neurons as the classes (67 for the facial recognition; 8 for the emotion recognition). The non-linearity for the last layer is set to be the soft-max function. The overall architecture of the first ConvNet is inspired by the network introduced by the Visual Geometry

Table 1 The tiny VGG

ConvNet configuration				
Input	224 × 224 × 3			
Conv	Receptive field	Stride	Padding	Feature map
	3 × 3	1	1	32
Conv	Receptive field	Stride	Padding	Feature map
	3 × 3	1	1	32
Maxpool	Filter		Stride	
	2 × 2		2	
Conv	Receptive field	Stride	Padding	Feature map
	3 × 3	1	1	32
Conv	Receptive field	Stride	Padding	Feature map
	3 × 3	1	1	32
Maxpool	Filter		Stride	
	2 × 2		2	
Conv	Receptive field	Stride	Padding	Feature map
	3 × 3	1	1	32
Conv	Receptive field	Stride	Padding	Feature map
	3 × 3	1	1	32
Maxpool	Filter		Stride	
	2 × 2		2	
FC	1024			
FC	512			
FC	256			
FC	No. of classes			
Softmax				

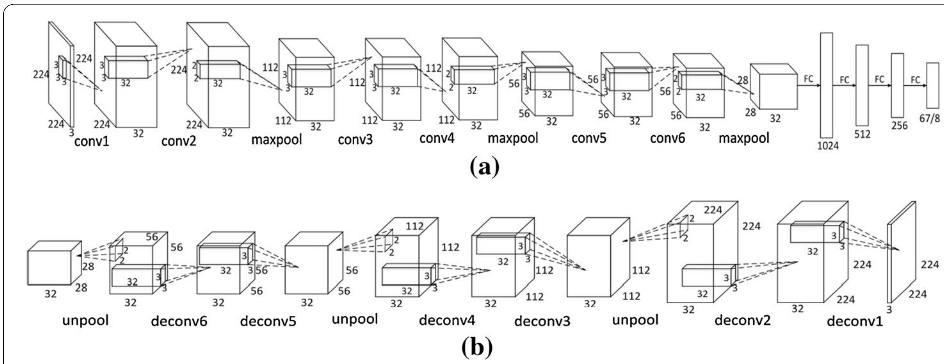

Fig. 3 Tiny VGG. **a** Convolution, **b** deconvolution

Group at the University of Oxford, known as the VGG net [4]. We then refer to our network as the Tiny VGG (Fig. 3a).

Deconvolution and the deconvolutional network

The process of reversing a convolution is generally referred to as deconvolution. This is achieved through deconvolutional layers. A deconvolutional layer utilizes the same receptive fields from the convolution layer that it is about to reverse. The fields are then flipped 180° horizontally and vertically. In some literature, the process of deconvolution is also referred as to the transposed convolution [19].

A deconvolution may result in up-samplings depending on how paddings are applied to the images. In other words, the critical part of performing a deconvolution (on a 2D image) is to ensure that the image should return to its size before the convolution. We have known that the equation for computing the size of a squared image after a convolution is given as:

$$i' = \frac{i - k + 2 \cdot p}{s} + 1 \tag{1}$$

where i is the size of the image before the convolution, k is the size of the receptive field, p is the number of padding used for the convolution, and s is the stride. Considering the different types of convolution:

1. Unit stride (stride is set to 1) with padding being 1.
2. Non-unit stride no padding.
3. Unit stride no padding.

The equations for computing the size of the image after the deconvolution, with respect to each of the aforementioned convolutions, are given as:

1. $i' + (k - 1) - 2 \cdot p$
2. $s \cdot (i' - 1) + k$
3. $i' + k - 1$

Figure 3b shows the deconvolutional network of the Tiny VGG, in which the unpooling layers are used to reverse the operations of maxpooling, where only the highest pixel values are kept for further use, resulting a down sized image depending on how large the filter is. Unpooling reverses maxpooling by plugging in the highest pixel values back into where they were and keeping other values as zeros, resulting an up sized image. Figure 4 (top) shows how unpooling works in theory, using a 2×2 filter with no overlapping. A

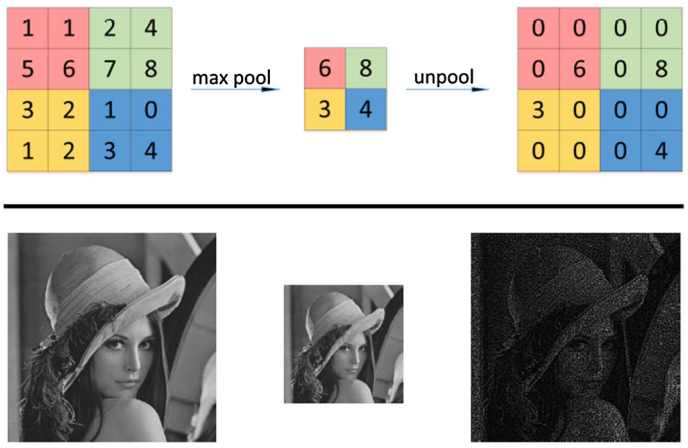

Fig. 4 Top: max pooling and unpooling in theory. Bottom: max pooling and unpooling in practice

practical example, which is based on a 512×512 grayscale image, is given in the bottom of the figure.

Preliminary evaluation

Training of the networks

The parameters for both networks are initialized using the approach proposed by Glorot and Bengio [20]. We found that using advanced optimizations, such as Ada [21] and Adam [22] updates, does not yield any performance improvement. Therefore, we use the stochastic mini-match decent to update the parameters, setting the size of a mini-batch to 12 and the learning rate to 0.01. The duration of the training process is set to be 100 epochs. To facilitate the training process, the network is trained on a Titan-X GPU, where approximately 20 images can be processed per second. The source code of this study are publicly available at: https://github.com/xingfang912/Understanding-deep-learning

Table 2 illustrates the testing accuracy achieved by the network with respect to the two classification tasks. Even though the task of emotion recognition has much fewer classes than the task of facial recognition, it turns out that the former task is more prone to over-fitting (Fig. 5).

Evaluations on miss-classified cases

For the facial recognition, the network has 59 miss-classified cases (17 in 45°, 19 in 90°, 23 in 135°), involving 37 subjects. Subject 48 is the most confusing person to the network, where the subject was miss-classified the most (four times).

Table 2 Testing accuracy

Facial recognition		Emotion recognition	
Training size	2412	Training size	2400
Testing size	2412	Testing size	2424
Classes	67	Classes	8
Accuracy	97.55%	Accuracy	90.97%

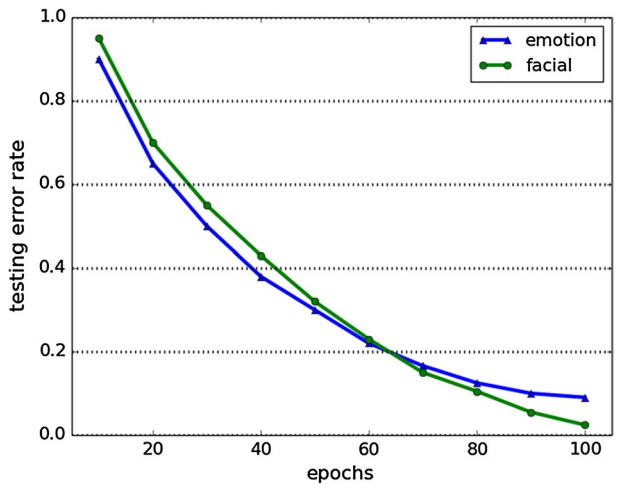

Fig. 5 Error rate of the network

For the emotion recognition, the network miss-identified 219 testing cases (73 in 45°, 70 in 90°, 76 in 135°) that include 56 subjects. The "sad" expression was missed the most, for 57 times, where it was identified as "neutral" for 31 times. The "surprised" expression was missed for only four times.

Analysis using backtracking and deconvolution

Motivation and the backtracking process

The preliminary evaluation only gives us a general idea towards the networks' performance. We are still far from having a complete understanding of the networks, especially regarding the mistakes that they made. Since performing deconvolutions on a trained CNN allows us to discover the features learned by the network, we are motivated to find out what are the facial features that can lead to the miss-classifications. This task can be achieved under the assistance of the backtracking process.

The backtracking process aims to track down the activated neurons from the output layer all the way back to the input of the first dense (FC) layer. Let a^{l-1} denote the activations in the $(l-1)$th dense layer. Assuming there are m neurons in this layer and n neurons in its next dense layer, l. Then a^{l-1} is a vector that has m real values and the weight matrix, W, between layer $l-1$ and layer l has the shape of $m \times n$. The back-tracking layer is designed based on the following observation: The activation in layer l at neuron k depends on the sum of the element-wise multiplication between a^{l-1} and the kth column of W.

To further explain the idea, let us consider the very last two layers of a trained CNN. Let the activation function in the $l-1$th layer be ReLU and let the lth layer be the output layer with the non-linearity set to be the soft-max function:

$$\frac{e^{x_k}}{\sum_j e^{x_j}} \tag{2}$$

Figure 6 is showing that the output layer indicates that the input data belongs to the kth class by having the maximum activated value, e^{x_k}, on its kth neuron. Since e^x is a monotonically increasing function, the value of e^{x_k} is fact determined by x_k, which it is computed as:

$$x_k = \sum_{i=1}^{m} a_i^{l-1} \cdot W_i^k + b_k \tag{3}$$

where a_i^{l-1} is the ith element in a^{l-1}; W_i^k is the ith element in the kth column of W; b_k is the kth value in b, which is the bias with respect to W. Since the biases in deep learning usually consist of small values, x_k is largely determined by the sum of the element-wise multiplication, $\sum_{i=1}^{m} a_i^{l-1} \cdot W_i^k$.

Because the values in a^{l-1} are either positive or zeros, due to the activation of ReLU, we claim that if the ith neuron is an activated neuron in the $(l-1)$th dense layer, it holds the property such that $a_i^{l-1} \cdot W_i^k > 0$.

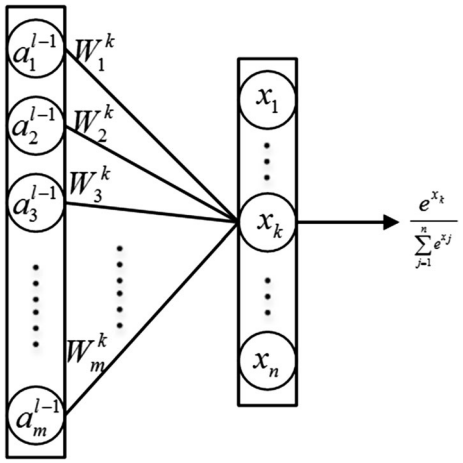

Fig. 6 Forward propagation of the last two layers of a CNN

The algorithm of backtracking is provided for performing more generalized backtracking processes with any given number of dense layers. The algorithm requires a trained neural network that has n FC layers before the output layer. $W_1, ..., W_n$ are the weight matrices for each of the FC layers, and W_o is the weight matrix of the output layer. The algorithm, at first, tracks down the activated neurons in the last FC layer. It then moves on to track such neurons in the previous FC layers. The algorithm finally outputs the indices of activated neurons. If the network is a CNN, the output of the algorithm is a set of pixel indices, where the pixels contain the features from the set of feature maps that determine the result of the classification.

Algorithm 1 The Backtracking Algorithm

Input: $W_1, ..., W_n$, and W_o
Output: L
Step 1:
Perform forward propagation to compute k, and $a_1, ..., a_n$
Set $L \leftarrow \emptyset$ and $L\_old \leftarrow \emptyset$

Step 2: Compute L_old
for the i^{th} row in W_o **do**
if $a_n[i] \cdot W_o[i][k] > 0$ **then**
$L\_old \leftarrow i$
end if
end for

Step 3: Compute L
for j from n to 1 **do**
for i in L_old **do**
for the r^{th} row in W_j **do**
if $a_j[r] \cdot W_j[r][i] > 0$ **then**
$L \leftarrow r$
end if
end for
end for
$L\_old \leftarrow L$ and $L \leftarrow \emptyset$
end for
return L_old as L

Visualization of the feature maps

Figure 7 uses the feature maps generated by the Tiny VGG network to demonstrate the effect of the backtracking process. The images in the figure are miss-classified testing cases from both classification tasks. For a better visualization purpose, we modify the pixel intensity values such that the black pixels are used for showing the facial features and the gray ones are used for the background.

The feature maps (the middle rows in Fig. 7) are in size of 28×28, indicating they are the feature maps serving as the input of the first FC layer of the network. Same feature maps are plotted in the bottom rows, where each of them has several missing pixels. Those missing pixels consist of the facial features, such as eyes, noses, mouths, eyebrows, and cheeks, which are discovered by the backtracking process. It is clearly to see that the feature maps in facial recognition have much more missing pixels than the ones in the emotion recognition. In fact, the average number of the missing pixels is 811 for the facial recognition and 188 for the emotion recognition. This fact suggests that there are more pixels (or facial features) contributed to the classification for the facial recognition than the emotion recognition.

Visualizing the features through deconvolution

In this subsection, we visualize the facial features that lead to the miss-classifications in both tasks. To do this, we apply the process of deconvolution, which provides top–down projections by mapping activations in feature maps generated in intermediate layers back to the input pixel space, showing the patterns that were learned by the feature maps.

The feature tracking algorithm provides a detailed description towards the visualization process. $D = \{d_i\}$ is a set of miss-classified images extracted from the testing set. The process outputs D', which is a set of images consisting facial features that lead to the miss-classifications. The backtracking algorithm is applied to obtain a L for d_i. To generate M'_i, it is crucial to save all the feature maps, M_i, which are used as the input of the

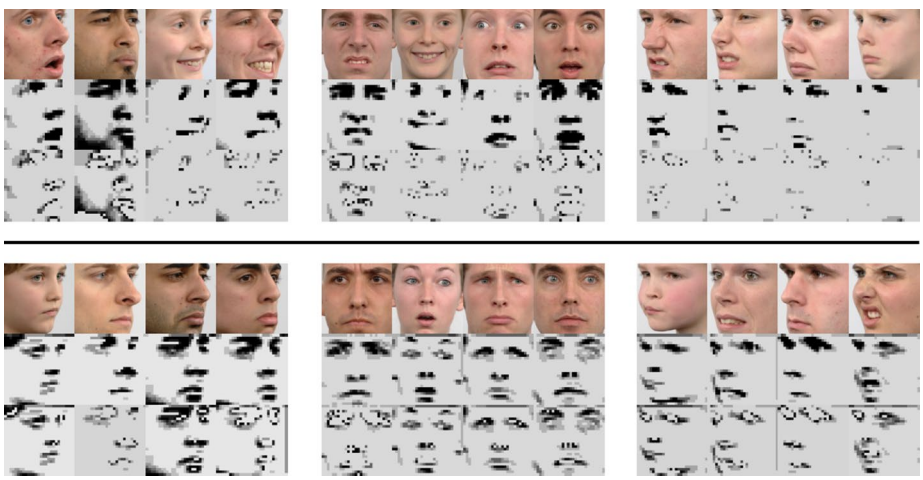

Fig. 7 Above the line: feature maps in facial recognition. Below the line: feature maps in emotion recognition. Top row: original images. Middle row: feature maps. Bottom row: feature maps with missing pixels

first FC layer of the CNN, generated during the forward propagation process. By suppressing the activations in m, it means the activations are replaced with values, such as zero. This will not allow the features that lead to miss-classifications to be mapped back to the input pixel space. Finally, we are able to obtain D' by computing the difference between dd_i and dd_i'.

Algorithm 2 The Feature Tracking Process

Input: a trained CNN and D
Output: $D^{'}$
  $D^{'} \leftarrow \emptyset$
  for d_i in D **do**
    apply the backtracking algorithm to obtain L
    $S \leftarrow \emptyset$
    for each feature map, m, in M_i **do**
      $m^{'} \leftarrow$ suppress the activations in m using L
      $S \leftarrow m^{'}$
    end for
    $M_i^{'} \leftarrow S$
    $dd_i \leftarrow Deconvolution(M_i)$
    $dd_i^{'} \leftarrow Deconvolution(M_i^{'})$
    $D^{'} \leftarrow |dd_i - dd_i^{'}|$
  end for
  return $D^{'}$

Visualization and analysis of the features

Figure 8 illustrates the features that are discovered by the feature tracking process. The same modification is applied onto the pixel intensity values for visualization. The 59 images are from the miss-classified cases in facial recognition. We can observe that the facial features leading to the miss-classifications are generally distributed over the entire faces, including eyes, nose, cheeks and mouth.

To examine the features in both correct and miss classifications, a comparative analysis is conducted based on two very similar images (Fig. 9) from the same person (subject

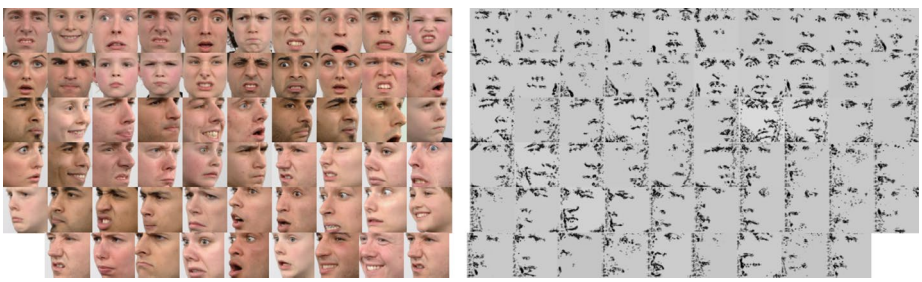

Fig. 8 Visualization of the features in facial recognition. Left: original images. Right: features

(a) **(b)**

Fig. 9 **a** Correctly classified case; **b** miss-classified case

40), with the only difference between the two images is the subject's gaze direction. The backtracking process is able to discover 623 and 693 activated pixels for Fig. 9a, b, respectively. And there are 547 pixels in common, in terms of their positions (pixel indices) in the feature maps. A closer observation (Fig. 10) identifies that the activation values of those pixels are largely different. To further understand how the pixels can affect the classification, we compare the results of the classifications with and without those pixels.

With the pixels, the results of the classification are: 0.99 on Fig. 9a under the class 40; 0.8 on Fig. 9b under the class 65 and 0.18 under the class 40. This indicates that the network has almost 100% of confidence that Fig. 9a should belong to subject 40 and it has 80% of confidence that Fig. 9b belongs to subject 65. To obtain the results without those pixels, we set their values to zero. In this case, the network's confidence with respect to Fig. 9a drops down to 0.8×10^{-5} under the class 40, which is nearly zero percent. Its confidence under the class 65 and the class 40 regarding Fig. 9b also drops to 0.0015 and 0.0006, respectively. In fact, after we apply the same analysis to the rest of the 58 images, we discover that common pixels largely determine results of the classifications. The activation values in those pixels are the decisive point in terms of the accuracy of the classification.

For the emotion recognition, a similar analysis is conducted based on the two groups of images (Fig. 11). Images in group A belong to the miss-classified cases; Images in group B are visually similar to the images in A but can be correctly classified by the network. The backtracking process is applied to both groups in order to identify the activated pixels. Unlike the situation in facial recognition, the common pixels do not contribute much for the miss-classifications. However, we discover that the activated pixels uniquely belonging to the feature maps of the images in group A can determine the classification results. In other words, those pixels are from the facial features leading to the miss-classifications. This can be proved that once their values are suppressed (setting to zero), the classifications will become correct. Table 3 lists the probability distributions of the eight cases in Fig. 11. By applying the feature tracking process based on those unique pixels, Fig. 11 also shows the facial features that lead to the miss-classifications.

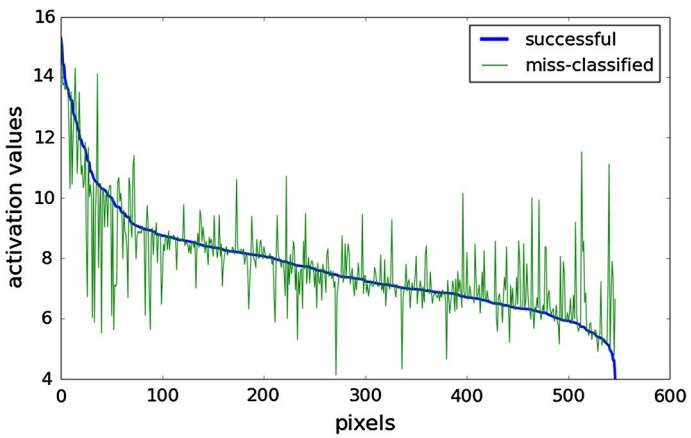

Fig. 10 The values of the common pixels

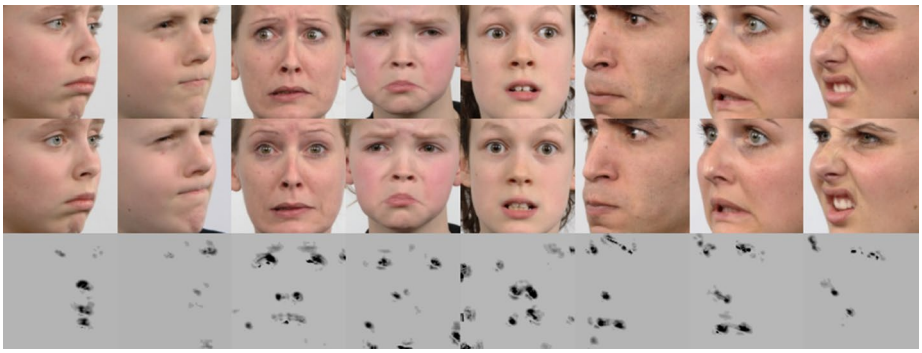

Fig. 11 Top: selected images from group A, emotions from left to right are miss-classified as: fearful, contemptuous, neutral, angry, disgusted, sad, surprised, happy; middle: selected images from group B; bottom: facial features that lead to the miss-classifications

Discussion and future work

In our early experiments, we did try to use the existing ConvNets such as the VGG-16 and the VGG-19. The training of the two networks fails on both classification tasks, due to the vanishing gradient problem. That is why we decided to reduce the number

Table 3 The distribution of confidences

	Probability							
	Angry	**Contemptuous**	**Disgusted**	**Fearful**	**Happy**	**Neutral**	**Sad**	**Surprised**
Case 1								
Before	6.36e−10	4.5e−7	6.71e−7	*9.89e−1*	2.59e−12	3.34e−3	7.27e−3	5.10e−4
After	5.57e−5	2.71e−4	5.57e−5	1.22e−1	5.57e−5	5.79e−3	*8.72e−1*	5.57e−5
Case 2								
Before	1.92e−1	*6.95e−1*	2.67e−6	3.66e−10	6.52e−10	3.17e−5	1.13e−1	1.39e−12
After	*9.12e−1*	4.56e−3	7.68e−7	7.45e−7	7.45e−7	1.72e−6	8.32e−2	7.45e−7
Case 3								
Before	2.70e−9	1.86e−7	7.16e−11	4.13e−1	1.26e−11	*5.87e−1*	2.41e−6	2.90e−9
After	1.32e−9	3.01e−9	1.32e−9	*9.99e−1*	1.32e−9	5.53e−5	1.84e−9	1.32e−9
Case 4								
Before	*9.75e−1*	5.47e−6	4.95e−10	7.94e−12	2.09e−17	6.16e−9	2.46e−2	1.43e−13
After	1.54e−2	6.97e−5	1.04e−7	1.04e−7	1.04e−7	2.99e−6	*9.85e−1*	1.04e−7
Case 5								
Before	6.89e−9	2.55e−6	*9.54e−1*	4.30e−2	8.70e−10	2.52e−3	4.87e−5	3.81e−7
After	9.58e−5	9.74e−5	1.14e−2	*8.08e−1*	9.58e−5	1.66e−1	1.35e−2	9.58e−5
Case 6								
Before	9.64e−2	7.11e−4	1.92e−10	2.00e−5	1.77e−14	2.17e−4	*9.03e−1*	4.44e−15
After	*9.87e−1*	7.78e−4	1.21e−8	1.26e−8	1.21e−8	4.12e−7	1.18e−2	1.21e−8
Case 7								
Before	2.08e−13	9.97e−9	5.66e−16	3.36e−1	1.29e−14	1.53e−1	2.37e−9	*5.11e−1*
After	5.13e−7	5.13e−7	5.13e−7	*8.98e−1*	5.13e−7	9.85e−2	5.13e−7	3.25e−3
Case 8								
Before	2.81e−12	1.02e−10	1.74e−1	1.65e−17	*8.26e−1*	8.52e−15	4.41e−14	1.39e−18
After	6.55e−10	6.55e−10	*9.79e−1*	6.55e−10	2.13e−2	6.55e−10	6.55e−10	6.55e−10

Italic values indicate probabilities of the miss-classified classes (before the suppression) and the probabilities of the correct classes (after the suppression)

of convolutional and max-pooling layers. We also avoid to use layers that do not yield visually appealing results in deconvolution, such as the average-pooling layer and the inception layer. We also explored some advanced numerical optimization methods like Adam and Adagrad, where we found the advanced methods were outperformed by the traditional stochastic gradient decent.

One possible future work is to apply data augmentation techniques onto the training data in order to improve the accuracy for emotion recognition. Especially, the augmentation will be focused on the facial features that lead to miss-classifications.

Conclusions

Convolutional neural networks have been widely adopted in solving problems in computer vision. Rather than focusing on proposing new network architectures, we decide to take a closer look at the miss-classified instances in order to gain a better understanding towards deep learning. Specifically, we firstly propose the back-tracking process that enables us to trace the activated pixels from the feature maps. We then propose the feature-tracking algorithm to visualize the facial features that lead to the miss-classifications. For the facial recognition, the activations of the common pixels are decisive for the result of classification. For the emotion recognition, the activations of the unique pixels indeed determine the result of classification.

Acknowledgements
Not applicable.

Competing interests
The author declares no competing interests.

Availability of data and materials
The Radbound Faces Database (RaFD) can be accessed via: http://www.socsci.ru.nl:8180/RaFD2/RaFD?p=main. The source code for this work is available at: https://github.com/xingfang912/Understanding-deep-learning.

Consent for publication
Not applicable.

Ethics approval and consent to participate
Not applicable.

Funding
Not applicable.

Publisher's Note

Received: 11 September 2017 Accepted: 3 November 2017
Published online: 13 November 2017

References

1. Lecun Y, Bengio Y, Hinton G. Deep learning. Nature. 2015;521(7553):436–44. https://doi.org/10.1038/nature14539.
2. Lecun Y, Bottou L, Bengio Y, Haffner P. Gradient-based learning applied to document recognition. Proc IEEE. 1998;86(11):2278–324. https://doi.org/10.1109/5.726791.
3. Krizhevsky A, Sutskever I, Hinton GE. Imagenet classification with deep convolutional neural networks. In: Pereira F, Burges CJC, Bottou L, Weinberger KQ, editors. Advances in neural information processing systems. vol. 25. Red Hook: Curran Associates, Inc.; 2012. p. 1097–105. http://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf. Accessed 10 Oct 2016.
4. Simonyan K, Zisserman A. Very deep convolutional networks for large-scale image recognition. 2014. arXiv preprint arXiv:1409.1556.
5. Szegedy C, Vanhoucke V, Ioffe S, Shlens J, Wojna Z. Rethinking the inception architecture for computer vision. 2015. arXiv preprint arXiv:1512.00567.

6. He K, Zhang X, Ren S, Sun J. Deep residual learning for image recognition. 2015. arXiv preprint arXiv:1512.03385.
7. Tulsiani S, Kar A, Carreira J, Malik J. Learning category-specific deformable 3d models for object reconstruction. IEEE Trans Pattern Anal Mach Intell. 2017;39(4):719–31. https://doi.org/10.1109/TPAMI.2016.2574713.
8. Mikolov T. Recurrent neural network based language model.
9. Sutskever I, Vinyals O, Le QV. Sequence to sequence learning with neural networks. In: Advances in neural information processing systems. 2014. p. 3104–12.
10. Sak H, Senior A, Beaufays F. Long short-term memory recurrent neural network architectures for large scale acoustic modeling. In: Fifteenth annual conference of the international speech communication association. 2014.
11. Cho K, Van Merriënboer B, Gulcehre C, Bahdanau D, Bougares F, Schwenk H, Bengio Y. Learning phrase representations using RNN encoder-decoder for statistical machine translation. 2014. arXiv preprint arXiv:1406.1078.
12. Bahdanau D, Cho K, Bengio Y. Neural machine translation by jointly learning to align and translate. 2014. arXiv preprint arXiv:1409.0473.
13. Cho SJK, Memisevic R, Bengio Y. On using very large target vocabulary for neural machine translation.
14. Fang X. Databridge: Bridging data using sociometric approaches. PhD thesis. 2016. Copyright—Database copyright ProQuest LLC; ProQuest does not claim copyright in the individual underlying works; Last updated—2016-08-05. http://libproxy.lib.ilstu.edu/login?url=https://search.proquest.com/docview/1808419308?accountid=11578.
15. Zeiler MD, Fergus R. Visualizing and understanding convolutional networks. 2013. arXiv preprint arXiv:1311.2901.
16. Zeiler MD, Krishnan D, Taylor GW, Fergus R. Deconvolutional networks. In: 2010 IEEE conference on computer vision and pattern recognition (CVPR). New York: IEEE; 2010. p. 2528–35.
17. Noh H, Hong S, Han B. Learning deconvolution network for semantic segmentation. In: 2015 IEEE international conference on computer vision (ICCV). 2015.
18. Langner O, Dotsch R, Bijlstra G, Wigboldus DH, Hawk ST, van Knippenberg A. Presentation and validation of the radboud faces database. Cogn Emot. 2010;24(8):1377–88.
19. Dumoulin V, Visin F. A guide to convolution arithmetic for deep learning. 2016. arXiv preprint arXiv:1603.07285.
20. Glorot X, Bengio Y. Understanding the difficulty of training deep feedforward neural networks. In: In proceedings of the international conference on artificial intelligence and statistics. Society for Artificial Intelligence and Statistics; 2010.
21. Duchi J, Hazan E, Singer Y. Adaptive subgradient methods for online learning and stochastic optimization. J Mach Learn Res. 2011;12(Jul):2121–59.
22. Kingma D, Ba J. Adam: a method for stochastic optimization. 2014. arXiv preprint arXiv:1412.6980.