Neural Networks 94 (2017) 159–172

Contents lists available at ScienceDirect

Neural Networks

journal homepage: www.elsevier.com/locate/neunet

Few-shot learning in deep networks through global prototyping

Sebastian Blaes [a], Thomas Burwick [a,b,*]

[a] Frankfurt Institute for Advanced Studies (FIAS), Goethe University Frankfurt, Ruth-Moufang-Str. 1, 60438 Frankfurt am Main, Germany
[b] Maastricht Centre for Systems Biology (MaCSBio), Maastricht University, P.O. Box 616, 6200 MD, Maastricht, The Netherlands

ARTICLE INFO

Article history:
Received 23 December 2016
Received in revised form 18 June 2017
Accepted 2 July 2017
Available online 24 July 2017

Keywords:
Convolutional Neural Networks
Object Recognition
Deep Learning
Few-Shot Learning
Transfer Learning

ABSTRACT

Training a deep convolution neural network (CNN) to succeed in visual object classification usually requires a great number of examples. Here, starting from such a pre-learned CNN, we study the task of extending the network to classify additional categories on the basis of only few examples ("few-shot learning"). We find that a simple and fast prototype-based learning procedure in the global feature layers ("Global Prototype Learning", GPL) leads to some remarkably good classification results for a large portion of the new classes. It requires only up to ten examples for the new classes to reach a plateau in performance. To understand this few-shot learning performance resulting from GPL as well as the performance of the original network, we use the t-SNE method (Maaten and Hinton, 2008) to visualize clusters of object category examples. This reveals the strong connection between classification performance and data distribution and explains why some new categories only need few examples for learning while others resist good classification results even when trained with many more examples.

1. Introduction

Given the recent success of Deep Convolution Neural Networks (CNNs) for visual object classification (reviewed, e.g., in LeCun, Bengio, and Hinton, 2015), the learning of additional categories based on only a few examples ("few-shot learning") is widely seen as an outstanding remaining challenge. Here, we study the few-shot learning task by using a pre-learned CNN as starting point. For the few-shot learning of additional new categories we then discuss a prototype-based approach that is straightforward and may even be seen as a close-to-minimal approach for a possible few-shot learning method. Despite the simple structure of this method, we find surprisingly good classification performance for the few-shot learning of a large portion of new object categories. To understand this performance, we use the t-SNE dimensionality reduction method (Maaten & Hinton, 2008) to explore the clustering of network activations in the classification space for different object categories. We find the cause of good and bad classification results to be linked to the positioning and distribution of the clusters relative to each other.

The discussed method is based on introducing prototypes in the global feature spaces of CNN final hidden layers. Accordingly, we refer to this learning method as Global Prototype Learning (GPL). It is inspired by vector quantization methods, most notably by the Growing Neural Gas (GNG) approach (Fritzke, 1995). The GNG construction of classification vectors (in the following also referred to as "classifiers" or "prototypes") is a few-shot learning procedure per se as it starts to have a new classifier at hand after the presentation of only one example which is not in the range of the old classifiers. In the following, we therefore imitate this construction of introducing a new classifier by identifying it with a new example and gradually improving it with next examples.

In Section 2, we refer to some earlier work on constructing new classifiers from only a few examples. There, we also give additional comments on the comparison with the GNG approach. In Section 3, we describe the CNN architecture which we use for our discussion. For concreteness, we use the network described in Chatfield, Simonyan, Vedaldi, and Zisserman (2014). In Section 4, we specify the few-shot learning procedure based on the GPL method to construct additional classifiers. In Section 5, as we want to compare GLP with related approaches, we describe a backpropagation-based alternative and a variant of GPL. In Sections 6 and 7, we analyze the resulting classification performance of GPL and the related approaches, respectively. There, we also apply the t-SNE visualization method (Maaten & Hinton, 2008) to study the connection between classification performance and data distribution.

* Corresponding author at: Frankfurt Institute for Advanced Studies (FIAS), Goethe University Frankfurt, Ruth-Moufang-Str. 1, 60438 Frankfurt am Main, Germany.
E-mail addresses: blaes@fias.uni-frankfurt.de (S. Blaes), burwick@fias.uni-frankfurt.de (T. Burwick).

http://dx.doi.org/10.1016/j.neunet.2017.07.001
0893-6080/

2. Earlier work

2.1. Learning from few examples

As the topic of learning new concepts with only a few training examples or no pre-labeled learning examples at all is one of the major challenges in deep learning, there has already been done quite a lot of research in that direction and the field is divided into different branches. There are mainly four major branches: Zero-shot learning, one- and few-shot learning and transfer learning.

Zero-shot learning has the most restrictive premise as no examples from the target domain are used during training. Instead, concepts are learned from related or even disjoint auxiliary source, for example, from large text corpora. Lampert, Nickisch, and Harmeling (2009) introduce an attribute-based classifier in which images are described by sets of high-level semantic per-class attributes. A semantic embedding of attributes is learned based on human knowledge. In the work of Jetley, Romera-Paredes, Jayasumana, and Torr (2015) , visual prototypical information is used to detect traffic signs and brand logos. Therefore, real world instances of objects are considered as imperfect realizations of their classes while classes are defined in form of their prototypes. (Note that precise understandings of the meaning of prototypes may vary; our definition in the context of the present work is given below in Section 4.1.) The idea is that during processing throughout the deep network most of the irrelevant information is discarded from the data of real world examples such that the high-level representation of input data is similar to the high-level representations of the prototypes. Zhang and Saligrama (2015) use the mixture of seen class proportion as a measurement for the similarity between unseen classes. Akata, Malinowski, Fritz, and Schiele (2016) combine an attribute-based classifier with a supervised semantic part annotation that allows to characterize objects by a set of attributes and their relative position to each other.

In one-shot learning (see, for example, Li, Fergus, and Perona, 2006 for a survey of the field), concepts are learned from a single example. Hence, target domains are usually of lower complexity as, for example, in case of the MNIST dataset (LeCun, Bottou, Bengio, & Haffner, 1998) of handwritten digits. Several works (Lake, Salakhutdinov, & Tenenbaum, 2013, 2015; Rezende, Mohamed, Danihelka, Gregor, & Wierstra, 2016) use probabilistic generative models for synthesis and analysis of handwritten digits. Others (Fink, 2005; Schroff, Kalenichenko, & Philbin, 2015) learn distance metrics from domains that are related to the target domain. The former implements a nearest neighbor classifier for the target domain that is based on the learned metric while the latter learns a mapping between image space and Euclidean space that translates similarities in faces into distances in the Euclidean space.

In transfer learning (see, for example, Pan & Yang, 2010 for a survey of the field), first a network is pre-learned on a more general dataset as the ImageNet (Deng et al., 2009) dataset and afterwards the classification network is relearned or fine-tuned on a complex but usually more specific target domain as bird species or urban tribes (Wang & Cottrell, 2015). Usually there exists a large number of training examples from the target domain but as the target domain is more specific variation within the training data is much lower. Apparently, this results in less expressive features in the earlier stages of the processing hierarchy that are outperformed by features learned from more general datasets (Donahue et al., 2014).

Few-shot learning allows for more than one but still a limited number of examples during training of new concepts. This allows for learning of complex concepts from complex data as the ones from the ImageNet dataset. In the work of Hariharan & Girshick (2016) a network is pre-learned from the ImageNet dataset (representational learning phase) and afterwards new classifiers are learned with only a few (between one and ten) examples (low-shot learning phase). During the low-shot learning phase only few example from the new classes but all examples from the old classes are available and new classes are learned simultaneously. Wang and Hebert (2016) study classifiers that are learned from thousands of examples and classifiers that are learned from only a few examples. It is hypothesized and demonstrated that there exists a "generic, category agnostic" transformation between the two models.

Recently, Hecht and Gepperth (2016) also proposed a deep network model that uses prototypes. In their approach prototypes are used to produce local activity maps in the first two hidden layers of a four layers deep network. A linear regression layer on top of the prototype layers is used for classification of instances from the MNIST dataset. In a loose sense their approach can be seen as complimentary to our approach since we use prototypes in the last layers of the processing hierarchy. In Hecht and Gepperth (2016) the model is studied especially with respect to memory consumption and training times which are compared to the memory consumption and training times in deep convolutional and shallow prototypical networks.

In the present work new classes are learned from a limited number of training examples in a network pre-trained on the ImageNet dataset. New classes are also learned from the ImageNet dataset. Hence, learned concepts as well as the data that is used for learning of new concepts are relatively complex and general. Classes are learned in succession one after the other. During learning of a new class only examples from that class but no examples from any other class are available. In that way 1000 new classes are added to the 1000 already existing classes.

2.2. Comparison with growing neural gas

As stated in the introduction, the method described in the following sections is inspired by the Growing Neural Gas (GNG) approach. For GNG, new examples are encoded as points in a feature space and similarity may be identified with Euclidean distances of these data points to sets of classifying vectors, also referred to as "prototypes". Whenever the minimal distance of a particular example to all prototypes is too large, this may be taken as an indication that the example is part of a new class and, therefore, this leads to the introduction of a new prototype. As the best guess for this new prototype is the example under consideration, the new prototype is identified with this example. Whenever next examples are presented and this new prototype turns out to be closest, the new examples are recognized as belonging to the new class and the prototype is also moved into the direction of the new example in order to take the additional information from this additional example into account.

In comparison to the described GNG approach, three modifications need to be applied. First, as we consider this approach in the context of a CNN, we stay with the paradigm of supervised learning. (Nevertheless, given the closeness to the self-organized GNG learning procedure, generalizations of our approach discussed to self-organized methods seem natural. Following such directions is, however, beyond the scope of the present work.) Second, for GNG the examples are encoded in a feature space that may be compared to the first layers of a CNN. Here, in contrast, we rather start from the other end by constructing the prototypes in the final layers of the CNN, that is, in the global feature layers which follow the convolutional layers. Third, we do not use Euclidean distance as the measure for similarity between examples and classifiers. Instead, as discussed in the following sections, in the context of the CNN a measure of directional similarity is appropriate.

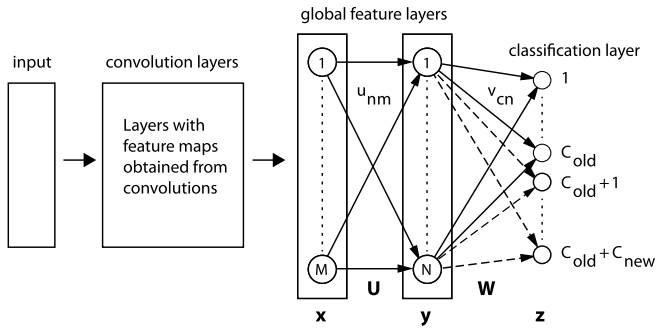

Fig. 1. The CNN architecture used in Section 4 for extending the classification to additional categories; see Chatfield et al. (2014) for more details of the network without the new classifiers. The dashed lines indicate these C_{new} new classifiers ("global prototypes") which are introduced through prototype-based learning (GPL) as described in Section 4. In Section 5.2, we also discuss couplings from second to last global feature layer (the one with output $\mathbf{x}$) to the classification layer; see Fig. 3.

3. Deep network architecture

The by-now standard CNN-Architecture for visual object recognition, pioneered in particular by Krizhevsky, Sutskever, and Hinton (2012), may be described as three stages of processing. First, given a certain input image representation, there is the convolution part of processing. Second, a network comprised of layers with all-to-all couplings (connecting all units from one layer to all units of the next) is subsequently included that combines all the inputs from the final convolutional layer. As all remaining retinotopy is lost at this stage, we may refer to these layers as global feature layers. In fact, much of the following concentrates on these layers, so that we introduce the notion of 'global feature layers' (GFLs) as an abbreviation to refer to this part of the network. Third, the output of the final layer of the GFLs is then classified by using classification vectors ("classifiers") that process this output to the classification layer, where using a softmax procedure may provide the classification results. As an illustration of the architecture described in the following see Fig. 1.

For concreteness, we now assume two GFLs and describe their outputs with the vectors $\mathbf{x}$ and $\mathbf{y}$, where the first GFL receives its input from the final convolutional layer and the second GFL receives its input from the first GFL. The subsequent classification layer receives its input from the final (the second) GFL and we refer to its output with the vector $\mathbf{z}$. (Here and in the following, bold face letters indicates vectors and matrices.) Assembling the corresponding weights between these layers in matrices $\mathbf{U}$ and $\mathbf{W}$, the processing is described with

$$\mathbf{z} = \mathbf{W}\mathbf{y} + \mathbf{b} \quad \text{and} \quad \mathbf{y} = \mathbf{g}(\mathbf{U}\mathbf{x} + \mathbf{d}), \tag{1}$$

where the $\mathbf{g}$ are the signal functions and $\mathbf{b}$, $\mathbf{d}$ are biases; see Fig. 1 for an illustration of the order of processing. We denote the dimensionalities of the vectors $\mathbf{x}$, $\mathbf{y}$, and $\mathbf{z}$ as, correspondingly, M, N, and C, where C is the number of classes. The signal function is chosen to be the standard ReLU function, that is, acting on a vector $\mathbf{a} = (a_1, \ldots, a_M)^\top$, the result is given by

$$\mathbf{g}(\mathbf{a}) = (\max(0, a_1), \ldots, \max(0, a_M))^\top. \tag{2}$$

We assume that C_{old} classes have been pre-learned in the network, using many examples for the backpropagation learning method. (In Section 6, we work with the network described in Chatfield et al. (2014) which is based on 1.2M training images for 1000 object categories.) Going beyond using this pre-learned network, we study the subsequent construction of classifiers for C_{new} additional classes that are constructed from only few examples, the restriction referred to as few-shot learning. Thus, there is a total of $C = C_{\text{old}} + C_{\text{new}}$ classes and we write

$$\mathbf{z} = \begin{bmatrix} \mathbf{z}_{\text{old}} \\ \mathbf{z}_{\text{new}} \end{bmatrix} = (z_1, \ldots, z_{C_{\text{old}}}, z_{C_{\text{old}}+1}, \ldots, z_{C_{\text{old}}+C_{\text{new}}})^\top, \tag{3}$$

where $\mathbf{z}_{\text{old}}$ and $\mathbf{z}_{\text{new}}$ are vectors of dimensions C_{old} and C_{new}, respectively. The weight matrix takes the form

$$\mathbf{W} = \begin{bmatrix} \mathbf{W}_{\text{old}} \\ \mathbf{W}_{\text{new}} \end{bmatrix} = (\mathbf{v}_1, \ldots, \mathbf{v}_{C_{\text{old}}}, \mathbf{v}_{C_{\text{old}}+1}, \ldots, \mathbf{v}_{C_{\text{old}}+C_{\text{new}}})^\top, \tag{4}$$

where the classification vectors ("classifiers") $\mathbf{v}_c$ have dimension N,

$$\mathbf{v}_c = (v_{c1}, \ldots, v_{cN})^\top, \tag{5}$$

so that $\mathbf{W}_{\text{old}}$ and $\mathbf{W}_{\text{new}}$ are matrices of dimension $C_{\text{old}} \times N$ and $C_{\text{new}} \times N$, respectively, and $\mathbf{W}$ is a $C \times N$ matrix (with $C = C_{\text{old}} + C_{\text{new}}$); see Fig. 1 for an illustration of this splitting into old and new classifiers. We refer to the N-dimensional space with coordinates $\mathbf{y}$ as "classification space". Correspondingly, also the bias splits into old and new components,

$$\mathbf{b} = \begin{bmatrix} \mathbf{b}_{\text{old}} \\ \mathbf{b}_{\text{new}} \end{bmatrix} = (b_1, \ldots, b_{C_{\text{old}}}, b_{C_{\text{old}+1}}, \ldots, b_{C_{\text{old}}+C_{\text{new}}})^\top, \tag{6}$$

constituting a vector of dimension C.

The confidences S_c are then obtained from applying the softmax function to $\mathbf{z} = (z_1, \ldots, z_C)^\top$, giving

$$S_c = \sigma_c(\mathbf{z}) = \frac{\exp(z_c)}{\sum_{c'=1}^{C} \exp(z_{c'})}. \tag{7}$$

The class c with largest S_c is the recognized class.

In Section 6, we restrict our discussion to a concrete pre-learned implementation described in Chatfield et al. (2014) with an implementation using MatConvNet (Vedaldi & Lenc, 2015). This implies that $C_{\text{old}} = 1000$ and $M = N = 4096$. Explicitly, we use the model version named VGG-F which consists of an RGB input layer of size $224 \times 224 \times 3$, five convolutional layers and three fully connected layers. The receptive fields of the first convolutional layer, given as num × size × size, have dimensions $64 \times 11 \times 11$. The receptive fields of the second layer have dimensions $256 \times 5 \times 5$ and the ones of the third to fifth layers $256 \times 3 \times 3$. The first and second fully connected layers consist of 4096 units while the third fully connected layer consists of 1000 units. The last fully connected layer is followed by a softmax layer that transforms the actives in the last layer into probabilities.

Our method to extend the network with C_{new} new classes is now specified in Section 4.

4. Global Prototype Learning (GPL) of new categories

4.1. New classifiers as prototypes in global feature space

The classifiers $\mathbf{v_c}$ of Eq. (5) are assumed to be pre-learned for $c = 1, \ldots, C_{\text{old}}$ where $C_{\text{old}} < C$. How do we choose the $C_{\text{new}} > 0$ new classifiers with indices $c = C_{\text{old}} + 1, \ldots, C$, where $C = C_{\text{old}} + C_{\text{new}}$?

For a reason that is stated below, we first want to split the classifiers $\mathbf{v}_c$ into two parts,

$$\mathbf{v}_c = \mathbf{v}_c^+ + \mathbf{v}_c^-, \tag{8}$$

with positive and negative components, respectively,

$$v_{cn}^+ = \max(0, v_{cn}) \quad \text{and} \quad v_{cn}^- = \min(0, v_{cn}), \tag{9}$$

for $n = 1, \ldots, N$. Assuming that we have E training examples ξ_e^c for each of the new classes $c, e = 1, \ldots, E, c = C_{\text{old}}+1, \ldots, C_{\text{old}}+C_{\text{new}}$,

we apply a corresponding separation to the inputs $\bar{\mathbf{y}}$ at the final global feature layer,

$$\bar{\mathbf{y}} = \mathbf{U}\mathbf{x} + \mathbf{b} = \bar{\mathbf{y}}^+ + \bar{\mathbf{y}}^-, \tag{10}$$

with

$$\bar{y}_n^+(\xi_e^c) = \max(0, \bar{y}_n(\xi_e^c)) \quad \text{and} \quad \bar{y}_n^-(\xi_e^c) = \min(0, \bar{y}_n(\xi_e^c)), \tag{11}$$

for $n = 1, \ldots, N$, where the separation in Eq. (10) depends on the particular input ξ_e^c to the network. The $\bar{\mathbf{y}}$ denotes the activity in the final global feature layer before applying the ReLU operation (compare with Eq. (1)). Note that for each n at least one of the components $\bar{y}_n^+$ and $\bar{y}_n^-$ is zero.

In the following, we want to study a choice for the new classifiers that is of particular simplicity,

$$\mathbf{v}_c = \mathbf{v}_c^+ + \mathbf{v}_c^- = \frac{\lambda_c^+}{E}\sum_{e=1}^{E}\bar{\mathbf{y}}^+(\xi_e^c) + \frac{\lambda_c^-}{E}\sum_{e=1}^{E}\bar{\mathbf{y}}^-(\xi_e^c), \tag{12}$$

for $c = C_{\text{old}} + 1, \ldots, C_{\text{old}} + C_{\text{new}}$, where $\mathbf{v}_c^+$ and $\mathbf{v}_c^-$ are given by the first and second term on the r.h.s., respectively. The $\lambda_c^\pm > 0$ are normalizations. The reason for splitting the definitions of $\mathbf{v}_c$ and $\bar{\mathbf{y}}$ into positive and negative parts is that we want to allow for different values of λ_c^+ and λ_c^-. This will be explained in Section 4.2. In the following, we refer to a vector given by Eq. (12) as global prototype, thereby indicating that it constitutes a combination of features represented in global feature space.

In case of $E = 1$, the new classifier is identified with the activation in the final global feature layer (before applying the ReLU truncation and normalized with $\lambda_c^\pm$) which is generated by the input image and, in this sense, we may describe this approach as a prototype-based learning. In case of several examples, $E > 1$, the global prototypes $\mathbf{v}_c$, with $c = C_{\text{old}}+1, \ldots, C_{\text{old}}+C_{\text{new}}$, are obtained from arithmetic averaging over the training example of each class. (The use of prototypes makes the method reminiscent of vector quantization; see de Vries, Memisevic, & Courville, 2016; Hecht & Gepperth, 2016 for other vector quantization inspired approaches to deep networks.)

What motivates the choice of Eq. (12)? Consider a particular input image ξ_e^c which is presented to the network, resulting in an activation $\bar{y}_n(\xi_e^c)$ in the pre-classification layer. If a response $\bar{y}_n(\xi_e^c)$ for some n is particularly large, this may be taken as an indication that the global feature encoded by unit n is particularly prominent in the input. Therefore, it is reasonable to take it as a characteristic for the new class c and give v_{cn} a large value, a relation expressed through Eq. (12).

Given that the old classifiers act on the activations $\mathbf{y}$ one may be tempted to construct the prototypes from $\mathbf{y}$ and use

$$\mathbf{v}_c' = \frac{\lambda_c}{E}\sum_{e=1}^{E}\mathbf{y}(\xi_e^c), \tag{13}$$

for $c = C_{\text{old}} + 1, \ldots, C_{\text{old}} + C_{\text{new}}$, a form that may be obtained from Eq. (12) by choosing $\lambda_c^+ = \lambda_c$ and $\lambda_c^- = 0$. Note, however, in general the old classifiers also have negative components. In contrast, using Eq. (13) would lead to new classifiers with only positive components as $y_n \geqslant 0$ for each n. This, however, is not advantageous: the negative components play an important role as argued in the following paragraph.

Consider that the applied input image leads to $y_n = 0$ for some n. This indicates that the global feature encoded by unit n is not present in or incompatible with the training example. We may then take this as a characteristic of the new class and seek an implementation that implies a decreased classification value z_c if the feature is present. As $y_n = 0$ is equivalent to $\bar{y}_n \leqslant 0$ this may again be implemented by relating v_{cn} to $\bar{y}_n \leqslant 0$ as done with Eq. (12). We may therefore expect that using Eq. (12) and not Eq. (13) leads to better classification results. Indeed, this is what we found with our simulations.

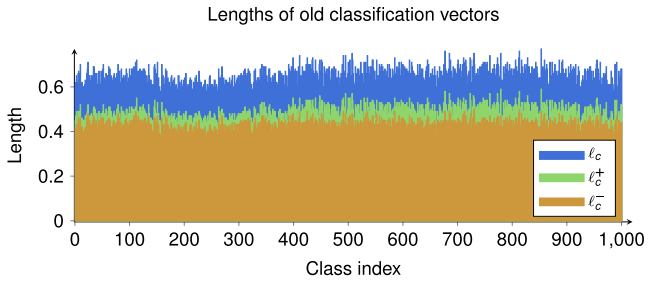

Fig. 2. Euclidean lengths of old classification vectors $\mathbf{v}_c$ (blue), of $\mathbf{v}_c^+$ (green) and $\mathbf{v}_c^-$ (yellow). See Eqs. (8) and (9) for the respective definitions. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

We still need to specify the new components of the bias vector $\mathbf{b}$ defined in Eq. (6). We found from simulations that setting the old components in the pre-learned network to zero does not substantially diminish its classification performance. Thus, given that the role of non-vanishing bias values does not seem to be essential, we decided to keep the construction simple by giving the new bias components vanishing values, $b_c = 0$ for $c = C_{\text{old}} + 1, \ldots, C_{\text{old}} + C_{\text{new}}$. For the old categories we keep the pre-learned non-vanishing values.

4.2. Relevance of classification vector length

The Euclidean length $\ell_c = |\mathbf{v}_c|$ of the classification vectors $\mathbf{v}_c$ is a property that is crucial for the classification result. Let us introduce $\mathbf{e}_c = \mathbf{v}_c/\ell_c$ so that $|\mathbf{e}_c| = 1$. Then

$$z_c = \ell_c \sum_{n=1}^{N} e_{cn} y_n \tag{14}$$

and it is obvious that ℓ_c has a decisive role for the recognition as class c. In fact, the length ℓ_c may even be seen as a bias towards the class c. In particular, Eq. (7) implies that $\ell_c \to \infty$ leads to $S_c \to 1$ and $S_{c'} \to 0$ for $c' \neq c$ if the sum in Eq. (14) is non-vanishing.

In Section 4.1, we mentioned the different roles that positive and negative components of the classifiers have to play. Correspondingly, we allow different values for the normalizations λ_c^+ and λ_c^- in Eq. (12). In that respect, it is of interest to compare the lengths of the $C_{\text{old}} = 1000$ pre-learned classifiers; see Fig. 2 for the values ℓ_c and $\ell_c^\pm = |\mathbf{v}_c^\pm|$ of the old classes. It turns out that each of these quantities has approximately the same length among the different pre-learned categories. Presumably, this is caused by each class having the same number of training examples so that no bias towards some classes was introduced. Note that this reflects a non-realistic assumption which entered the learning process: in nature, not all classes have the same frequentness.

We may then define the average values of the old categories,

$$\ell^\pm = \frac{1}{C_{\text{old}}}\sum_{c=1}^{C_{\text{old}}}|\mathbf{v}_c^\pm|, \tag{15}$$

and choose the $\lambda_c^\pm$ for the new categories such that $|\mathbf{v}_c^\pm| = \ell^\pm$ for the new classifiers, $c = C_{\text{old}} + 1, \ldots, C_{\text{old}} + C_{\text{new}}$. As we want to avoid any bias in favor or against the new classes, this gives the new classifiers a length that is of the same order as the corresponding values for the old classes.

Let us also introduce two alternative choices of $\lambda_c^\pm$ for the new classes which correspond to maximal and minimal values of the

old classes. Thus, we define

$$\ell^{\pm}_{\max} = \max_{1\leqslant c\leqslant C_{\text{old}}} |\mathbf{v}^{\pm}_c| \quad \text{and} \quad \ell^{\pm}_{\min} = \min_{1\leqslant c\leqslant C_{\text{old}}} |\mathbf{v}^{\pm}_c|, \tag{16}$$

and introduce two alternative choices for the $\lambda^{\pm}_c$ of the new categories such that $|\mathbf{v}^{\pm}_c| = \ell^{\pm}_{\min}$ or $|\mathbf{v}^{\pm}_c| = \ell^{\pm}_{\max}$ for $c = C_{\text{old}} + 1, \ldots, C_{\text{old}} + C_{\text{new}}$. In Section 6, we consider alternative new classifiers with such maximal and minimal normalizations for comparison with the choice given by $|\mathbf{v}^{\pm}_c| = \ell^{\pm}$. This will illustrate the relevance of the normalization and confirm the appropriateness of choosing some intermediate value $|\mathbf{v}^{\pm}_c| = \ell^{\pm}$ for the new categories.

5. Related approaches

Although there exists already a large number of very different approaches to solve few-shot learning (see Section 2 for a list) the problem remains a challenge and is still considered unsolved. Hence, it is difficult to pick out and compare new ideas against a certain realization that could serve as a base line model. For this reason, we decided to compare GPL against two other approaches. The first one is an adaptation of the backpropagation algorithm and allows to compare GPL with a learning method that is closely related to the learning method used during the initial training of the network. The second one uses additional features from earlier layers in the final classification step.

In Section 5.1, we use backpropagation to optimize the choice of the prototype. We refer to the corresponding backpropagation-based learning method as normalized backpropagation learning (NBL) and compare NBL with GPL. In Section 5.2, we comment on another variant of the GPL procedure. This one uses a classification space that is extended to include not only the final global feature layer (the one with output $\mathbf{y}$) but also the second to last (the one with output $\mathbf{x}$). The corresponding extended global prototype has components in this extended classification space; see Fig. 3.

5.1. Prototype optimization through backpropagation

The old classifiers $\mathbf{v}_c$, with $c = 1, \ldots, C_{\text{old}}$, have been learned through backpropagation of errors (Chatfield et al., 2014). It may then be of interest to compare GPL with some alternative learning procedure that uses backpropagation also for the construction of the new classifiers. Note, however, a crucial restriction that we applied to the construction of the new classifiers $\mathbf{v}_c$, with $c = C_{\text{old}} + 1, \ldots, C_{\text{old}} + C_{\text{new}}$. We assumed that no examples of the foregoing classifiers are still available when the new classifier is constructed. Correspondingly, we do not allow for any adaption of previously learned weights. The learning of the foregoing classifiers is assumed to be finished. Note that the notion of foregoing classifiers does not only refer to the old classifiers but to any $\mathbf{v}_{c'}$ with $1 \leqslant c' < c$ if a new category c with $C_{\text{old}} < c \leqslant C$ is learned (as stated above, we assume that the new categories are learned in succession). Thus, if we want to compare GPL to a backpropagation-based alternative, we have to make sure that this procedure is defined with the same restriction to make a comparison valid.

Given the mentioned restriction, any backpropagation learning cannot develop the same power as in case of the simultaneous pre-learning of the old categories. There a parallel balancing of the different classification errors is possible, while here, given only the examples of a new to-be-learned category, backpropagation is restricted to optimize the classifiers $\mathbf{v}_c$ of the new category c.

Let us then define the backpropagation-based optimized search for the new classifiers with the following three characteristics. First, the error function for class c is chosen to be softmax log-loss, the same that was used for the learning of the C_{old} original classes

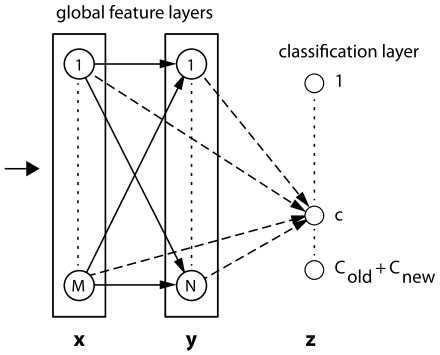

Fig. 3. Going beyond the architecture shown in Fig. 1, we now also allow for couplings from the second to last global feature layer (the one with output $\mathbf{x}$) to the new classification units, illustrated for one of these units. The input and convolutional part are as in Fig. 1 (not repeated here). See the discussion in Section 5.2.

(Chatfield et al., 2014). Second, for each new class c a classification vector $\mathbf{v}_c$ is added to the classification layer and, in accordance with the mentioned restriction, during backpropagation training all weights except for the weights of the new classification vector are kept frozen. Third, after each epoch $\mathbf{v}^{+}_c$ ($\mathbf{v}^{-}_c$) is rescaled such that its length equals ℓ^{+} (ℓ^{-}) as defined in Eqs. (15). As discussed in Section 4.2, this normalization is introduced to avoid overfitting with respect to the new class because of a too long classification vector. Motivated by this property, we refer to the described learning procedure as normalized backpropagation learning (NBL). Effectively the normalization property implies that the NBL only determines the direction of the new classification vector while the length is set correspondingly to how it is set in case of GPL.

5.2. Using additional features in the classification

In Section 6.4 we will discuss that some of the new classes are difficult to learn because they have large overlaps with similar old classes. One approach to solve the problem with the non-separable classes might be to include additional features into the classification. For example, in Section 6.4 we demonstrate that the categories crackers, french loafs and bagels are difficult to distinguish. The features needed to distinguish such classes may no longer be present at the final global feature layer but may still be given at earlier layers. The idea is therefore to use features from earlier layers in hope that some of these earlier features are still able to differentiate between, for example, crackers, french loafs and bagels but are consolidated in later layers as there was no relevant information in the feature space that is given by those features for the classification of the old classes.

To this end, we modify the network architecture that was described in Section 3. We extend the network architecture to allow also for couplings from the second-last global feature layer to the classification units; see Fig. 3. Thus, Eqs. (1) are replaced by

$$\mathbf{z} = \begin{bmatrix} \mathbf{z}_{\text{old}} \\ \mathbf{z}_{\text{new}} \end{bmatrix} = \begin{bmatrix} \mathbf{W}_{\text{old}} & \mathbf{0} \\ \mathbf{W}_{\text{new}} & \mathbf{W}'_{\text{new}} \end{bmatrix} \begin{bmatrix} \mathbf{y} \\ \mathbf{x} \end{bmatrix} + \begin{bmatrix} \mathbf{b}_{\text{old}} \\ \mathbf{b}_{\text{new}} \end{bmatrix} \quad \text{and} \quad \mathbf{y} = \mathbf{g}(\mathbf{U}\mathbf{x} + \mathbf{d}), \tag{17}$$

where the weight matrix has dimension $C \times (N + M)$. It may be written in terms of the classifiers

$$\begin{bmatrix} \mathbf{W}_{\text{old}} & \mathbf{0} \\ \mathbf{W}_{\text{new}} & \mathbf{W}'_{\text{new}} \end{bmatrix} = (\mathbf{v}_1, \ldots, \mathbf{v}_{C_{\text{old}}}, \mathbf{v}_{C_{\text{old}}+1}, \ldots, \mathbf{v}_{C_{\text{old}}+C_{\text{new}}})^{\top}, \tag{18}$$

where the classifiers $\mathbf{v}_c$ are now $(N + M)$-dimensional vectors,

$$\mathbf{v}_c = (v_{c1}, \ldots, v_{cN}, v_{c(N+1)}, \ldots, v_{c(N+M)})^{\top}, \tag{19}$$

for $c = 1, \ldots, C$. As we do not want to touch the classification of the old categories, we set

$$v_{cm} = 0 \quad \text{for } c = 1, \ldots, C_{\text{old}} \quad \text{and} \quad m = N+1, \ldots, N+M. \tag{20}$$

in Eq. (18). The sub-matrix $\mathbf{W}_{\text{old}}$ is given by the pre-learned network and the components of $\mathbf{W}_{\text{new}}$ are defined as in Eq. (12), replacing only $\lambda_c^{\pm} \to \lambda_c'^{\pm}$ to allow for a different normalization (specified with the remarks following Eq. (23)). For the bias $\mathbf{b}$ we chose the same values as described in Section 4.1. What remains to be specified are the additional components given by $\mathbf{W}'_{\text{new}}$.

In analogy to Eqs. (10) and (11), we refer to the input at the first global feature layer as $\bar{\mathbf{x}}$ and split this input into positive and negative parts:

$$\bar{\mathbf{x}} = \bar{\mathbf{x}}^+ + \bar{\mathbf{x}}^-(\mathbf{x} = \mathbf{g}(\bar{\mathbf{x}})), \tag{21}$$

where the separation into the components

$$\bar{x}_m^+(\xi_e^c) = \max(0, \bar{x}_m(\xi_e^c)) \quad \text{and} \quad \bar{x}_m^-(\xi_e^c) = \min(0, \bar{x}_m(\xi_e^c)), \tag{22}$$

for $m = 1, \ldots, M$, depends on the particular input ξ_e^c to the network (see Section 4.1). In analogy to Eq. (12), we then define $\mathbf{W}'_{\text{new}}$ by the components of the new classifiers that connect to the first global feature layer through

$$v_{c(N+m)} = v^+_{c(N+m)} + v^-_{c(N+m)} = \frac{\lambda_c'^+}{E} \sum_{e=1}^{E} \bar{x}_m^+(\xi_e^c) + \frac{\lambda_c'^-}{E} \sum_{e=1}^{E} \bar{x}_m^-(\xi_e^c), \tag{23}$$

for $c = C_{\text{old}} + 1, \ldots, C_{\text{old}} + C_{\text{new}}$ and $m = 1, \ldots, M$, while the first N components of the new classifiers $\mathbf{v}_c$ are again given by Eq. (12). The $\mathbf{v}_c^+$ and $\mathbf{v}_n^-$ components are defined in terms of the first and second sum on the r.h.s., respectively. As in Section 4, we define the normalizations $\lambda_c'^{\pm}$ such that $|\mathbf{v}_c^{\pm}| = \ell^{\pm}$ for the new categories, $c = C_{\text{old}} + 1, \ldots, C_{\text{old}} + C_{\text{new}}$.

As this version of constructing new classifiers uses an extended form of the GPL of Section 4, we refer to it in the following as "extended GPL" (ext-GPL).

6. Results

To discuss the described GPL procedure in the light of a concrete example, we now use as a starting point a pre-learned CNN with $C_{\text{old}} = 1000$ object categories (Chatfield et al., 2014) (for more details see the last paragraph in Section 3). Using GPL to realize few-shot learning, we extend the network with C_{new} classifiers. In Section 6.1, we specify the chosen datasets for training and tests. In Section 6.2, the resulting performance of the GPL procedure is presented for $C_{\text{new}} = 1$ and up to $C_{\text{new}} = 1000$. For a more specific and illustrative discussion, three representatives are picked out among the new classes that show intermediate (walrus), good (sphinx) and poor (cracker) classification performances. Section 6.3 studies how good the old classes that where seen during the initial training can be relearned with GPL. Section 6.4 links the classification performances of different classes to different data distributions in classification space. This is done by comparing the three specified classes in relation to other classes through using the t-SNE visualization method (Maaten & Hinton, 2008).

6.1. Datasets for the few-shot learning and tests

To evaluate the classification performance of old and new classes learned with GPL, we constructed training and test sets for all the $C_{\text{old}} = 1000$ old classes and up to $C_{\text{new}} = 1000$ new classes from the ImageNet dataset (Deng et al., 2009). The ImageNet dataset consists of more than 100,000 object classes with on average 1000 images per class. Classes in the ImageNet dataset are hierarchically ordered based on their WordNet (Miller, 1995) id. Training, validation and test examples for the initial training were obtained from the same dataset; hence, complexity of old and new classes as well as the used training examples are comparable. New classes are randomly drawn from the ImageNet dataset and added to the pre-trained network under the following conditions: A potential new class (1) is not already among the old classes, (2) is not among the already chosen new classes, (3) is not a (direct or indirect) child class (in the WordNet hierarchy) of an old or new class, (4) has no old or new class as a child class. To avoid that a single new class blocks too many other potential new classes it is only added if (5) it has not more than 20 child classes. All new classes are learned in succession and each individual class is learned only with positive examples from that class and no negative examples from any other class.

We use cross-validation to study and test the few-shot learning procedure. That is, we construct, for all of the 1000 old and each of the 1000 possible new categories, $K = 5$ training sets $\mathcal{T} = \{\mathcal{T}_1, \ldots, \mathcal{T}_K\}$ with $|\mathcal{T}_k| = t = 20$ images and test sets $\mathcal{S} = \{\mathcal{S}_1, \ldots, \mathcal{S}_K\}$ with $|\mathcal{S}_k| = s = 150 - 20 = 130$ images, $k = 1, \ldots, K$, from a base set of 150 examples in total per class. Following this, for each class K times $t + s$ images are randomly drawn from the corresponding base set such that $\mathcal{T}_k \cap \mathcal{S}_k = \varnothing$ for $k = 1, \ldots, K$.

In order to be able to compare the performance of newly learned classes with the performance of the initially learned classes, the publicly accessible ILSVRC2012 validation set (Russakovsky et al., 2015) is used as test set for the initially learned classes.

6.2. Performance of new classifiers constructed with GPL

6.2.1. Adding one new class ($C_{new} = 1$)

To begin with, we study the case $C = 1001$, that is, $C_{\text{new}} = 1$ and compare the individual errors for all 1000 possible new categories (from the dataset described in Section 6.1) with the errors of the $C_{\text{old}} = 1000$ old categories (before the new class is added to the network); see Panels A and B of Fig. 4. (The effect of including multiple new categories on the classification performance of the old and new categories is discussed in Section 6.2.3 and in the context of Fig. 7. In case of only one new category the effect is negligible as discussed at the end of this section.) Top-1 errors are defined as the fraction of examples from the test sets (as defined in Section 6.1) that are not classified as the correct class. Top-5 errors are defined as the fraction of examples from the test sets where the correct class is not among the five classes with the highest confidence, that is, the five classes with the highest probabilities after the soft-max operation.

While old classes perform generally better, both old and new classes show a broad spectrum of performances. Panels C and D of Fig. 4 compare the corresponding histograms of errors. Old classes have been learned simultaneously with the backpropagation algorithm (Chatfield et al., 2014). This algorithm aims to minimize the average prediction error of all classes. Hence, there are only few classes that perform exceptional good or bad. The distribution of top-1 errors from old classes (blue bars in panel C) is centered around a mean value of 0.42 with a standard deviation of 0.19. The distribution of top-5 errors (blue bars in panel D) has a mean value of 0.19 and is even more narrow with a standard deviation of 0.13. The distributions of top-1 and top-5 errors from new classes (yellow bars in panels C and D, respectively) that are learned from 10 training examples per class are much broader and resemble almost uniform distributions. However, for the top-1 and top-5 errors there are notable overlaps between the distributions of the new and old class errors (gray areas in panels C and D) which

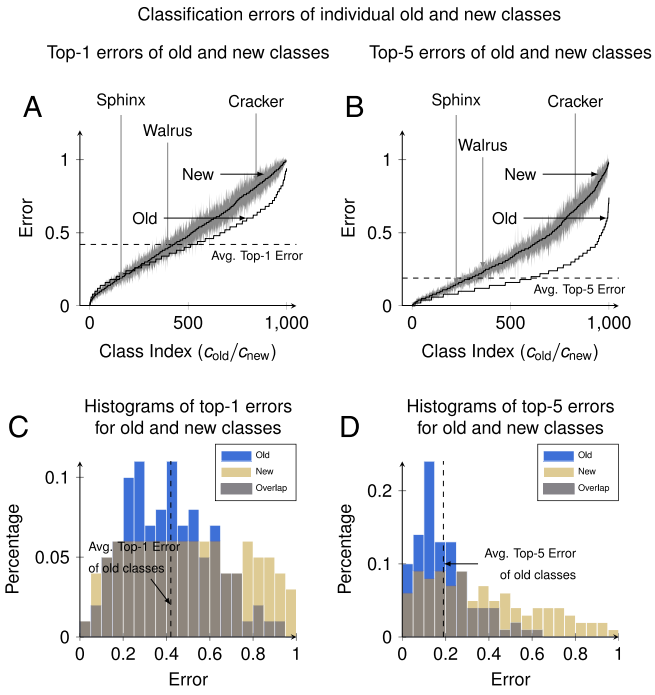

Fig. 4. $C_{new} = 1$. Panels A and B show a comparison between individual errors of all the $C_{old} = 1000$ old classes and the $C_{new} = 1000$ possible new classes. Indices c_{old} and c_{new} are chosen so that the errors can be shown in increasing order. New classes are learned from 10 training examples. Dashed lines in the left and right panels indicate the average top-1 and top-5 errors, respectively, of the old classes in case that $C_{new} = 0$. Vertical arrows point at the locations of three different new classes: walrus, sphinx, cracker. These are typical among all new classes and are discussed in more detail in Section 6.2.2. Errors of new classes are average values obtained via cross-validation (for details see Section 6.1). Lighter gray bands around curves in panels A and B indicate the standard deviation. In this case, old classes are static; hence, errors of old classes are fixed. Panels C and D show the corresponding histograms of errors from old and new classes. Counts are transformed into rates by dividing the individual counts by the number of old or new classes, respectively. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

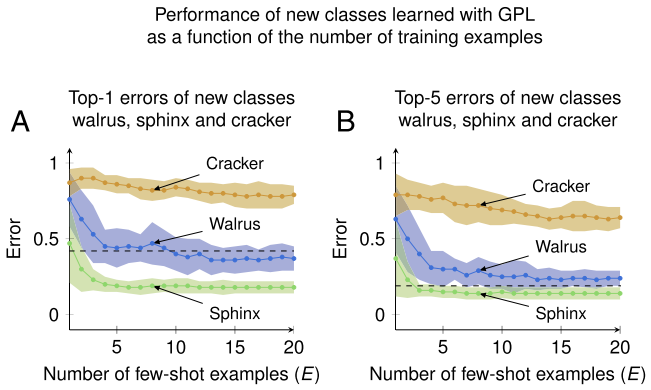

Fig. 5. $C_{new} = 1$. Classification error rates for the walrus (blue), sphinx (green) and cracker (yellow) classes as a function of the number of examples used during learning. Dots connected with lines show the average error rates after K repetitions with different training/test sets (as explained in Section 6.1). Lighter color bands around lines indicate the standard deviation. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

amount to more than 80% in case of the top-1 errors. These overlaps indicate that the average performance of new classes is comparable to the average performance of old classes.

Remarkably, a substantial portion of the new classifiers perform as good as or better than the average old classes although only few (10) training examples enter their construction. To better understand the origin of this observation, we now take a closer look at three of the new classes in Section 6.2.2. These are chosen to have similar (walrus), better (sphinx) and worse (cracker) performance in comparison with the old classifiers; see the corresponding marks in Fig. 4. Using these classes will allow to link the different performances to different kinds of data distributions in the classification space as discussed in Section 6.4.

Another important question is how the 1001st class affects the performance of old classes. We can measure the effect by the number of examples from all the old classes that are mistakenly classified as the 1001st class. In case of the new walrus class and using 50 examples per old class (50,000 examples in total) in 0.03% (± 0.01%) of the cases the 1001st class has the highest confidence among the 1000 old and the one new class. In 0.12% (± 0.05%) of the cases the 1001st class is among the 5 classes with the highest confidence. See Table 1 for more details.

6.2.2. Three typical new classes

Three of the new classes (walrus, sphinx, cracker) are highlighted in panels A and B of Fig. 4. These classes perform very differently and are typical examples among all new classes. The reasons for the performance gap between these three classes are rooted in the data itself and the network's capability to cluster the data in the classification space (the space with coordinates $\mathbf{y}$). Only if instances of one class are well separated from instances of all the other classes in the classification space the network is able to correctly classify most of the instances of that class. This will be discussed in Section 6.4.

Before we discuss the relation to data distributions, however, we want to use the three concrete new classes to demonstrate two other aspects of the GPL procedure: the actual dependency of the performance on the number of training examples (Section 4.1) and the relevance of choosing an appropriate vector length for the new classifiers (as discussed in Section 4.2).

Panels A and B of Fig. 5 show the top-1 and top-5 error rates for the classes walrus (blue), sphinx (green) and cracker (yellow) as a function of the number of training examples used during learning with the GPL algorithm. As new classes are learned independently from each other, in the following we assume that each new class is the $(C_{old} + 1) = 1001$st class in the network. In this case $|\mathbf{v}^+_{1001}|$ ($|\mathbf{v}^-_{1001}|$) is rescaled to ℓ^+ (ℓ^-) (see Eqs. (9) and (15) for the respective definitions). A justification for this choice will be given later in this section. Using 10 examples during training, the walrus class approaches a top-1 classification error rate of 0.40 (±0.11) and a top-5 error rate of 0.25 (±0.08). The walrus class performs as good as the average old class (dashed lines). The sphinx class is an example of a class that can be learned exceptionally good. Already with 10 training examples the top-1 error rate of the sphinx class approaches a value of 0.19 (±0.05) while the top-5 error rate approaches a value of 0.15 (±0.04). The cracker class, on the other hand, is an example for a class that is very hard to learn. With 10 training examples the top-1 error rate reaches a value of 0.84 (±0.06) and the top-5 error rate a value of 0.69 (±0.10). Even with 20 training examples the error rates do not drop significantly lower.

In case of the sphinx class and to some degree in case of the walrus class one prototype might already be enough to produce a classification vector that gives suitable classification results. After one training example the top-1 error rate of the sphinx class amounts to 0.47 (0.26) which is already close to the average top-1 error of the old classes. As a general rule, error rates settle already after approximately five to 10 examples and even with more examples error rates do not drop significant lower. This implies that only very few training examples are sufficient to learn new classes in a

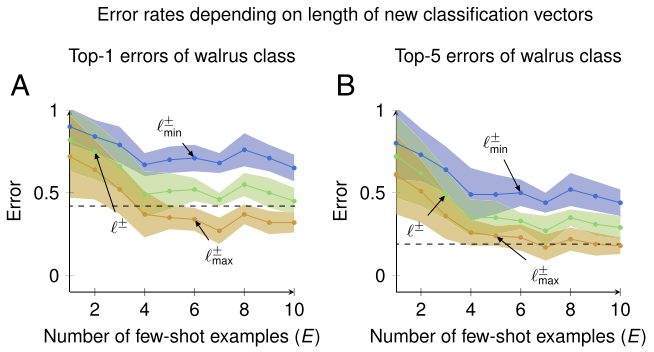

Fig. 6. $C_{\text{new}} = 1$. Panels A and B show the classification error rates of the walrus class as a function of the number of training examples and for three different lengths of its classifier $\mathbf{v}_c^+$ ($\mathbf{v}_c^-$) (see Eq. (9) for the respective definitions). Blue curves show the error rates for a length of $\ell^{\pm}_{\text{min}}$, green curves for a length of $\ell^{\pm}$ and yellow curves for a length of $\ell^{\pm}_{\text{max}}$. Lighter color bands around lines indicate the standard deviation after K repetitions with different training/test sets (as explained in Section 6.1). See Eqs. (15) and (16) for the respective definitions of $\ell^{(\pm)}$, $\ell^{(\pm)}_{\text{min}}$, and $\ell^{(\pm)}_{\text{max}}$. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

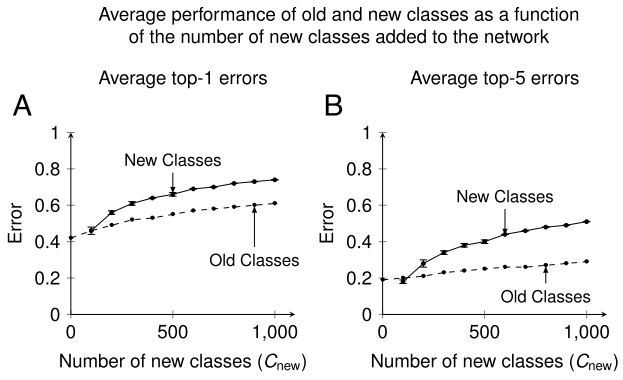

Fig. 7. $1 \leqslant C_{\text{new}} \leqslant 1000$. Average top-1 (A) and top-5 (B) errors of old (dashed lines) and new (solid lines) classes as a function of the number of new classes added to the network. $C_{\text{new}} = 0$ (dashed lines) marks the average network error if no new classes are added to the network. To test the isolated performance of new classes networks with only C_{new} classifiers for different sets of new classes are constructed.

Table 1
Number of times an example of the validation set for the old classes is misclassified as the walrus class (learned with GPL) for three different lengths of the classification vector.

Length	Top-1	Top-5
$\ell^{\pm}_{\text{min}}$	3.8 ± 2.5	23.4 ± 14.8
$\ell^{\pm}$	14.4 ± 5.9	63.8 ± 27.9
$\ell^{\pm}_{\text{max}}$	44.4 ± 21.5	167.4 ± 52.2

network that is pre-trained with a general dataset as the ImageNet dataset.

Fig. 6 shows the top-1 and top-5 classification error rates of the walrus class as a function of the number of training examples and for three different lengths of $\mathbf{v}^+_{1001}$ ($\mathbf{v}^-_{1001}$): ℓ^+_{min} (ℓ^-_{min}), ℓ^+ (ℓ^-) and ℓ^+_{max} (ℓ^-_{max}) (see Eq. (16) for the definitions of $\ell^{\pm}_{\text{min}}$ and $\ell^{\pm}_{\text{max}}$, respectively). As one would expect (see Section 4.2), the error rate is directly correlated with the length of the classification vector. With $\ell^{\pm}_{\text{min}}$ as the length of $|\mathbf{v}^{\pm}_{1001}|$ and 10 training examples the top-1 error reaches a value of 0.65 (± 0.08). In case of $\ell^{\pm}_{\text{max}}$ and the same number of training examples the top-1 error rate drops as low as 0.32 (± 0.06). However, the lower error rate of the 1001st class comes with a trade-off concerning all the other classes. Table 1 shows the number of examples of the validation set for old classes that are mistakenly classified as the walrus class in case that the walrus class is learned with the GPL algorithm. Classifying images of this set as one of the new classes directly affects the performance of the old classes.

To be considered as good, a classification vector should not only correctly classify the majority of examples from the own class but should also mistakenly classify as few examples as possible from all the other classes. Both conflicting properties are directly affected by the length of the classification vector. Hence, it is important to find a compromise that is balanced in both ways. For $\ell^{\pm}_{\text{min}}$ as the length of $|\mathbf{v}^{\pm}_{1001}|$ 3.8 (± 2.5) examples of the validation set for the old classes are mistakenly classified as the walrus class and in 23.4 (± 14.8) instances the walrus class is among the five classes with the highest confidence for images from the validation set of the old classes. In case of $\ell^{\pm}_{\text{max}}$ already 44.4 (± 18.5) examples of the old classes' validation set are classified as walrus class and in 167.4 (± 52.2) instances the walrus class is among the five classes with the highest confidence. If only one new class is added to the network the effect does not seem to be too dramatic (only 0.08% of the examples of the old classes' validation set are classified as the walrus class) but if more classes are added the introduced error adds up and can have a significant effect on the overall network performance (see the respective curves in Fig. 7). A good compromise for the length of $\mathbf{v}^{\pm}_{1001}$ seems to be $\ell^{\pm}$ as it produces good results concerning the quality of the new classes without affecting other classes too much.

6.2.3. *Adding many new classes* ($1 \leqslant C_{new} \leqslant 1000$)

Fig. 7 shows the average error of old and new classes as a function of the number of new classes added to the network. To test the isolated performance of new classes for every tested C_{new} multiple networks with only $C = C_{\text{new}}$ classifier are constructed from different sets of C_{new} new classes. The data point at $C_{\text{new}} = 0$ (dashed lines) shows the network performance in case that no new classes are added to the network (performance of the original network).

Adding more new classes to the network increases the errors of old as well as new classes as each new class that is added to the network might result in a misclassification of examples from other classes. This is shown for the walrus class by Table 1. However, there is no sudden breakdown of the overall network performance as more new classes are added to the network such that new classes can be added in a controlled manner.

6.3. *Relearning of old classes*

It is also interesting to see how the old classes perform if they are relearned with GPL. The reasoning behind this test is as follows. During the initial training the network learned filters that are specifically tailored to support an optimal separation of instances from the different old classes in the global feature space. Therefore, the center of data clouds defined by all possible realizations of the different classes should have great distances to each other making it relatively easy for GPL to find vectors that achieve good classification results. Given the small amount of training examples in the context of this work and the possible sizes of class distributions in the global feature space the question remains how good an optimal class vector can be approximated with GPL. To test this issue we proceed as follows. First, we delete the entire classification layer and relearn all the new classes with GPL. For relearning of old classes we use 10 examples per class that are randomly picked from the base sets (150 examples per class) obtained from the ImageNet dataset (see Section 6.1 for details). In Section 6.2.2 it is shown that after 10 examples learning usually reaches a plateau in performance. Panel A of Fig. 8 shows the dot product $\mathbf{v}_c^T \mathbf{v}'_c$ for $c = 1, \ldots, C_{\text{old}}$ between the original class vectors $\mathbf{v}$ and the

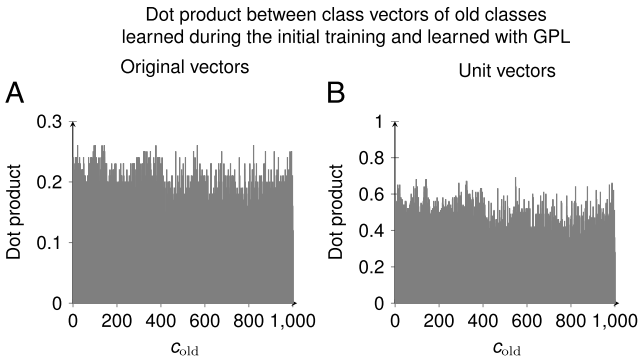

Fig. 8. Dot products between class vectors of old classes learned during the initial training with the backpropagation algorithm and relearned with GPL. Panel A shows the dot product for the original vectors. Panel B shows the dot product between the vectors after they are renormalized to unit length.

class vectors $\mathbf{v}'$ that are learned with GPL. If we take the original class vector as a quasi optimal classifier for the old classes this quantity measures how similar the class vectors learned with GPL are compared to these quasi optimal classifiers. As the dot product of two vectors depend on the length of the individual vectors a better quantity to measure the similarity between the class vectors is $\mathbf{v}_c^T\mathbf{v}_c'/(|\mathbf{v}_c| \cdot |\mathbf{v}_c'|)$. Panel B of Fig. 8 shows this quantity for all old classes. There is notable difference between the classifiers learned during the initial training and learned with GPL. This is also reflected in the individual errors of old classes relearned with GPL. These are shown in Fig. 9 as well as the histograms of errors of old classes initially learned with the backpropagation algorithm (blue) and old classes relearned with GPL (yellow). This figure can be seen in direct comparison to Fig. 4 which compares the errors of old classes initially learned with the backpropagation algorithm with the error of entirely new classes learned with GPL. As one would expect, old classes are relearned with GPL slightly better than entirely new classes. However, GPL performance, especially with the rather strict constrains imposed on the learning, is not as good as the backpropagation algorithm in the initial training.

6.4. Clustering in the classification space

In the following, we want to qualitatively relate the error rates observed in Fig. 4 to the data distributions of the different classes. For concreteness, we restrict the discussion to the three typical classes chosen in Section 6.2.2. We study their classification performances by relating these to data distributions in the classification space. Be reminded that the classification space is N_G = 4096-dimensional (with coordinates $\mathbf{y}$). These distributions may be visualized by, first, using a representational dataset for each of the studied classes and then, second, giving a two-dimensional visualization of these datasets. As datasets for the new classes, we use subsets of 50 examples obtained from the base sets (with 150 images) which were defined in Section 6.1. Moreover, we compare these distributions with representations of old classes as explained in the next paragraph. For the two-dimensional visualization we use the t-SNE method (Maaten & Hinton, 2008).

For each of the mentioned typical new classes, we determine the two most similar old classes. This is done as follows. For each of the 150 images from the base set of the new class, the top-5 classification results are determined among the old classes. The most similar old class is then defined to be the class that most often appears among these 150 top-5 winners. Analogously, the second most similar classes are the ones that have the second rank in this competition. Thereby, we find that the new walrus, sphinx, and cracker classes have as most similar old classes the stupa, sea

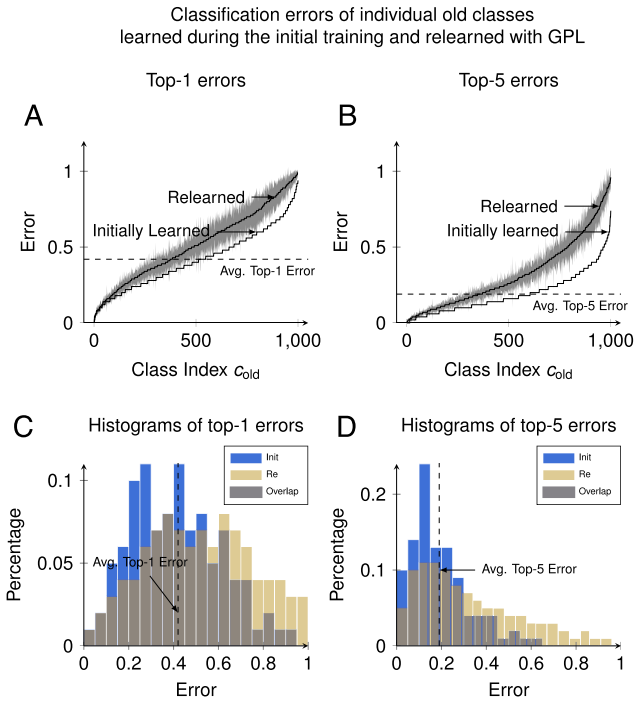

Fig. 9. Panels A and B show a comparison between individual errors of all the $C_{\text{old}} = 1000$ old classes learned during the initial training and relearned with GPL. Indices c_{old} are chosen so that the errors can be shown in increasing order. Old classes are relearned from 10 training examples. Dashed lines in the left and right panels indicate the average top-1 and top-5 errors, respectively, of the old classes learned during the initial training. Panels C and D show the corresponding histograms of errors from old classes learned during the initial training (Init) and relearned with GPL (Re). Counts are transformed into rates by dividing the individual counts by the number of old or new classes, respectively. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

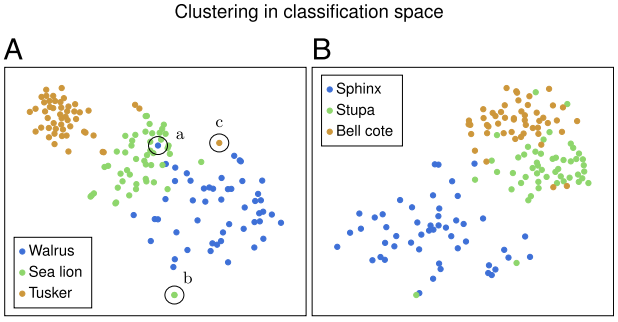

Fig. 10. (A) clustering of 50 instances of the new walrus class as well as instances of the two most similar old classes in classification space (the definition of similarity is given in Section 6.4): 50 examples of the sea lion and 50 examples of the tusker class. (B) clustering of 50 instances of the sphinx class together with its two most similar old classes, 50 examples of the stupa class as well as 50 examples of the bell cote class. For illustration purposes, the embedding of data points in the 4096-dimensional classification space is plotted in a *two*-dimensional space that is obtained via dimensionality reduction with the t-SNE procedure. The outliers marked by the circles a, b, c are generated by the images shown in Fig. 11 and are discussed in Section 6.4. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

lion, and french loaf classes, respectively. The second most similar old classes are, correspondingly, tusker, bell cote, and bagel. The datasets that we use to represent the old classes are the validation sets (50 images) used to determine the classification performance plotted in Fig. 4.

Fig. 10(A), (B), and 12 show the resulting visualizations of the data distributions for the new walrus, sphinx, and cracker classes,

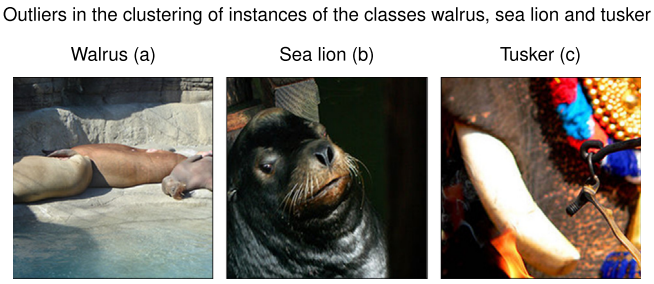

Fig. 11. Inputs that correspond to the outliers highlighted in panel A of Fig. 10. Left: an instance of the new walrus class. Middle: an instance of the old sea lion class. Right: an example of the old tusker class.

respectively, together with their two most similar old classes. By inspection, we find a confirmation for the statement that the quality of performance in classification is related to the relative positioning of the data clouds. The walrus class with its average classification performance shows a data cloud that is well separable from the two most similar classes, the sea lion and the tusker; see Fig. 10(A). It is reasonable to expect that this is at the root of the rather good performance of the new classifier: it corresponds to the average performance of the old categories which were pre-learned with many more examples.

This judgment is supported by comparing the other two classes. The sphinx class with its even superior performance shows even more separated data clouds; see Fig. 10(B). Correspondingly, as shown in Fig. 4, the sphinx classification has an error rate (resulting from the GPL low-shot learning) clearly below that of the average pre-learned categories (resulting from the backpropagation learning with many more examples).

Given these observations, it is then not surprising that the class that shows a rather high error rate in Fig. 4, the cracker class, has a data cloud that is not found to be as separable as the walrus and the sphinx classes. The cracker data cloud mixes strongly with the two similar classes, the french loaf and bagel classes; see Fig. 12.

At this point, it may also be of interest to take a closer look at the kind of images that lead to the distributions shown in the discussed figures. In particular, we consider certain regions that appear in the distribution of the walrus class as this class shows an average classification performance. Moreover, we consider regions of the distribution of the cracker class as this class fails to realize a good classification performance.

In Fig. 10(A), despite the separability of the data clusters, there are also outliers and it is instructive to see the corresponding input images; see Fig. 11. In general, all three clusters are well separated which is also reflected in the classification errors. As the blue curves in panels A and B of Fig. 5 show the walrus class performs as good as the average old class. Outliers result primarily from ambiguous input data. For example the three walruses in the left image of Fig. 11 are mostly depicted from behind and the one animal on the right side even misses the tusks. Hence, the walruses are mistaken as sea lion. The sea lion in the middle image is only shown partly such that the typical body form is barely recognizable. Lastly, only a tusk is shown in the image on the right.

In contrast to the walrus and sphinx classes, the instances of the cracker and its two most similar classes in Fig. 12 do not form well separated clusters. Instead, there is a large overlap of features that appear in instances of all three classes. Hence, examples from different classes may be more similar to each other than examples from the same class which eventually results in the more mixed distribution of data points than in the cases of the walrus and sphinx classes.

Six groups of three data points are highlighted in Fig. 12, three labeled with numbers and three labeled with letters. Groups that

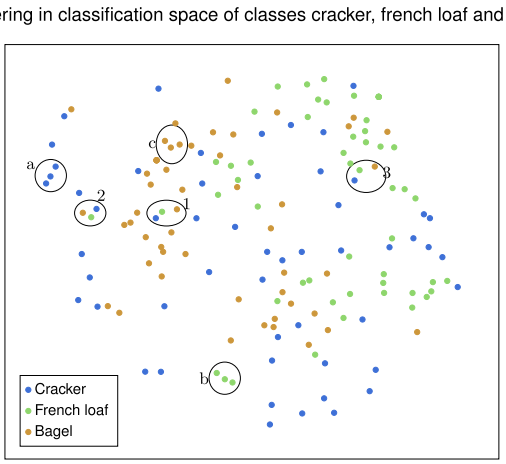

Fig. 12. Clustering of 50 examples from the new cracker class as well as examples of the two most similar old classes in classification space: 50 examples from each of the french loaf and bagel classes. Six groups of three data points are highlighted where all data points within one group are in close proximity and, therefore, share many similar features. One set of groups is labeled with numbers 1–3 and contains data points from different classes while the other set of groups is labeled with letters a to c and contains data points from the same class. These instances originate from images shown in Figs. 13 and 14, respectively. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

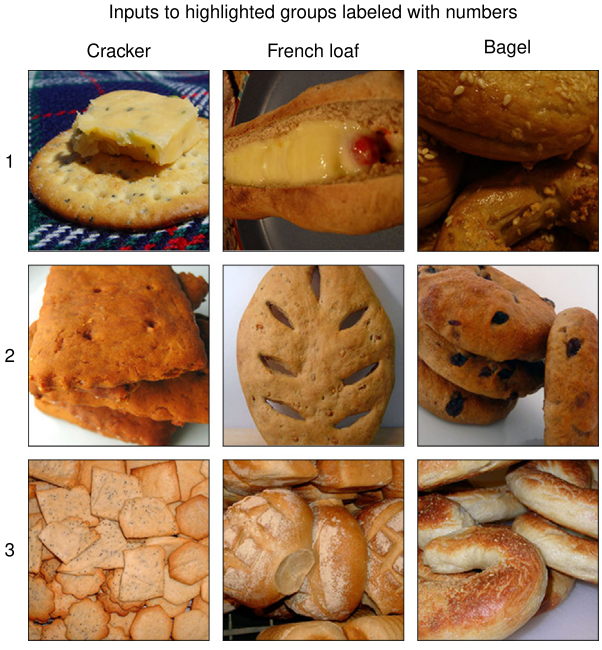

Fig. 13. Input images that correspond to the highlighted groups of examples that are labeled with numbers 1, 2, 3 in Fig. 12.

are labeled with numbers 1–3 contain three data points where each data point belongs to a different class. Corresponding input images are shown in Fig. 13. Despite from the fact that the data points belong to different classes all of them share a set of similar features; hence, their close proximity in the low dimensional visualization space of the classification space. Using human inspection, one might expect that the following common features of the input images led to the proximity in the representation. (Of course, a more reliable conclusion would need additional inquiries.) The images corresponding to group 1 in Fig. 12, displayed in the first

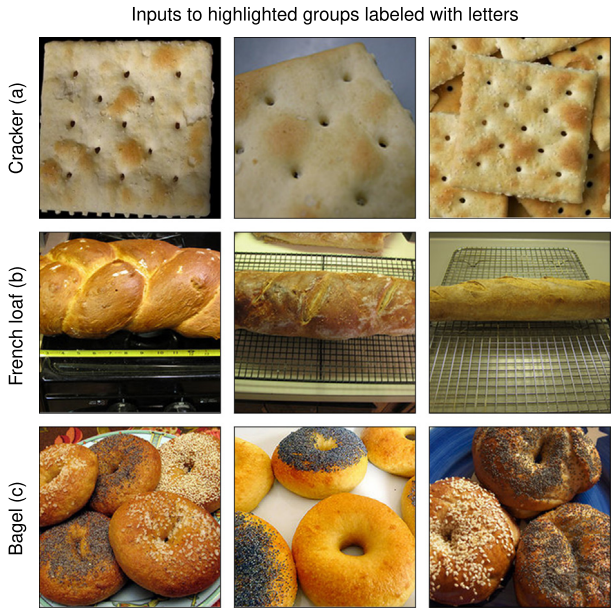

Fig. 14. Input images that correspond to the highlighted groups of examples that are labeled with letters a, b, c in Fig. 12.

row of Fig. 13, show some pronounced horizontal structure: the shadow of the cracker, the opening of the french loaf, the boundary between two bagels. All objects in the second row of Fig. 13, corresponding to group 2 in Fig. 12, have multiple smaller holes in it and especially the french loaf and bagel are very atypical for their respective classes. Because of the angle at which the bagels are pictured, the typical big hole in the middle of them is barely recognizable and even humans might be tempted to identify the objects as some kind of cracker. The group 3 in Fig. 12, stemming from the images displayed in the third row of Fig. 13, shares the general color pallet and all inputs depict multiple objects. Such similarities in the input images might cause the proximity in classification space that is present in Fig. 12.

The groups in Fig. 12 which are labeled with letters a to c contain data points that belong to the same class. Again data points from the same group have largely overlapping feature sets because of their close proximity. Corresponding inputs are shown in Fig. 14. Especially group 'a' hints to one possible solution for the 'cracker problem'. This group of inputs depicts a very special kind of cracker. Therefore, instead of looking at the group of crackers as one homogeneous group it might be more beneficial to split problematic classes in subclasses of groups that separates them from other groups in the parent class. Ideally this process runs in a self-organized manner and without or with only minimal supervision. Approaches of self-organized learning were already proposed in the context of growing neural gas (GNG) (Fritzke, 1995) and generalized learning vector quantization (GLVQ) (Sato & Yamada, 1996). Recently, de Vries et al. de Vries et al. (2016) applied GLVQ to deep convolutional networks and showed that the classical softmax layer at the end of the processing hierarchy can be replaced by a classifier based on GLVC that behaves better especially in cases of outliers. Combining this approach with GNG might be the subject of possible future research efforts in order to solve the classification problem that shows up in the context of the cracker class.

The visualizations presented in this section confirms that the performance of individual new classes depend on the capability of the network to cluster instances of different but similar classes in distinct and well separated clusters and the data used for training itself. If there is a large overlap between the relevant features of one

Table 2
Number of times an example of the validation set for the old classes is misclassified as the walrus class (learned with NBL) for three different lengths of the classification vector.

Length	Top-1	Top-5
$\ell^{\pm}_{\min}$	8.6 ± 8.6	249.4 ± 77.9
$\ell^{\pm}$	81.8 ± 11.9	320.8 ± 100.9
$\ell^{\pm}_{\max}$	56.6 ± 15.6	541.6 ± 106.0

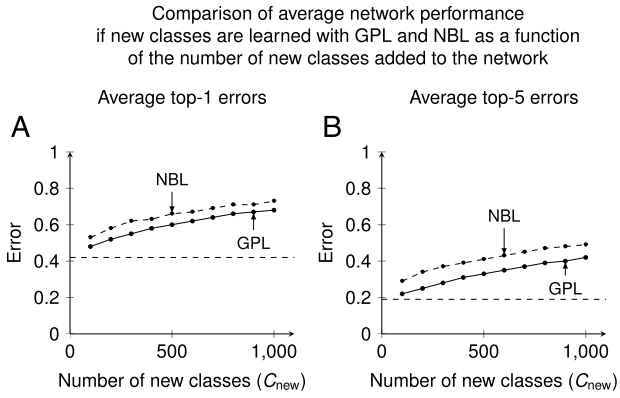

Fig. 15. $1 \leqslant C_{\text{new}} \leqslant 1000$. Average top-1 (A) and top-5 (B) network errors as a function of the number of new classes added to the network for GPL and NBL. The mean network error is defined as the average error among the $C_{\text{old}} = 1000$ old and a random set of C_{new} new classes.

class and the relevant features of other similar classes it might be very difficult for the network to distinguish between these classes even if many more training examples are used. Thus, even the failure of the low-shot learning for some new classes is found not to originate in the use of only few examples but instead in the non-separable data distribution of such classes in the classification space.

7. Comparison with related approaches

7.1. Prototype optimization through backpropagation

In Section 5.1 we introduced an alternative approach to learn new classes from few examples that is based on the backpropagation algorithm. We implemented such a NBL method with the following additional details. Initial weights are drawn from a standard normal distribution and scaled with a factor of 0.01. 30 epochs are used for training. During the first five epochs a learning rate of 5×10^{-4} is used followed by a learning rate of 2.5^{-4} during the next 10 epochs and finally a learning rate of 1.25×10^{-4} is used during the final 15 epochs.

However, this comes with a generally higher misclassification rate of examples from other classes as it is indicated for the walrus class in Table 2. While in case of GPL and $\left|\mathbf{v}^{\pm}_{1001}\right| = \ell^{\pm}$ 14.4 ($\pm$5.9) examples are misclassified as walrus class (see Table 1), in case of NBL 81.8 ($\pm$11.9) examples are misclassified as walrus class. This directly affects the overall network performance as it is shown in Fig. 15 where the average network errors are notably higher in case of NBL than in case of GPL.

In the light of the decreased overall network performance of the NBL method as displayed in Fig. 15, the improved performance for particular classes appears to show overfitting. Thus, the NBL would need to be complemented with the choice of a stopping criterion. So far, given the results of Fig. 15 no improvement of overall performance by using NBL instead of GPL may be concluded. What remains is the computational disadvantage of NBL resulting from

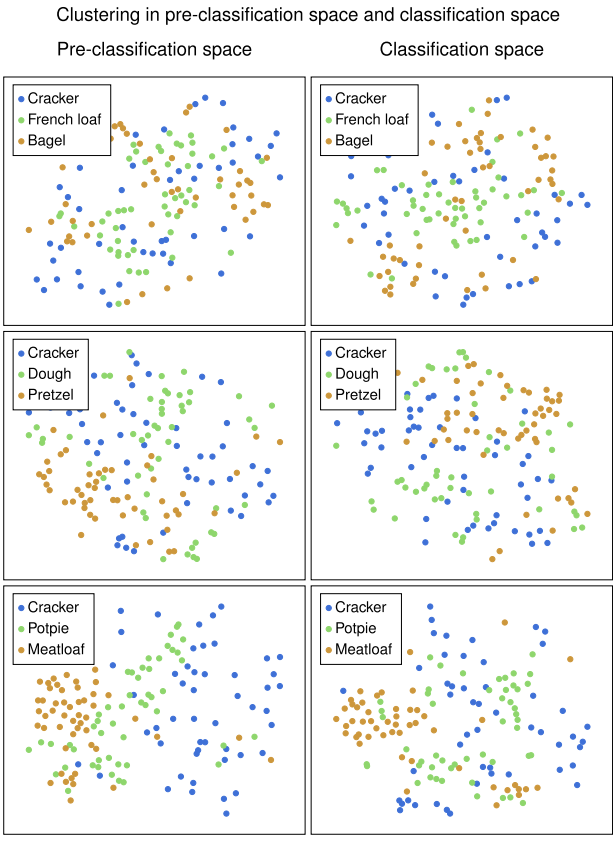

Fig. 16. Two-dimensional visualization of the clustering of examples from the cracker class in (left column) the $M = 4096$-dimensional pre-classification space given by the coordinates $\mathbf{x}$, and (left column) the $N = 4096$-dimensional classification space, given by the coordinates $\mathbf{y}$. Additionally, the embeddings of instances of the six most similar old classes are shown (the notion of similarity was defined in Section 6.4).

the need to use several incremental backpropagation (gradient descent) steps while the GPL learning implements the new classifier through the straightforward non-incremental computation given by Eq. (12).

7.2. Using additional features in the classification

We now consider the extension of GPL (ext-GPL) introduced in Section 5.2. This method uses the global features encoded at the second-to-last global feature layer; see Fig. 3. In Section 6.4, we defined the notion of most similar old categories. Here, we compare the cracker category with its six most similar categories. These are, in the order or decreasing similarity, the following categories: French loaf, bagel, dough, pretzel, potpie, meatloaf. Generalizing the visualizations of Section 6.4, we now apply the t-SNE visualization method to the layer with coordinates $\mathbf{y}$ (classification space) and also to the second-to-last global feature layer, the layer with coordinates $\mathbf{x}$ (referred to as pre-classification space); see Fig. 16 for the result.

With a view on Fig. 16, it is obvious that the mixing of distributions of the different classes which we linked to the poor performance of the cracker classifier in Section 6.4 is not overcome by including the pre-classification space. There is substantial overlap also in the pre-classification space. The average error rates for applying the ext-GPL method as introduced in Section 5.2 are plotted in Fig. 17. We find that there is an improvement in the classification performance of the cracker classifier, the new class

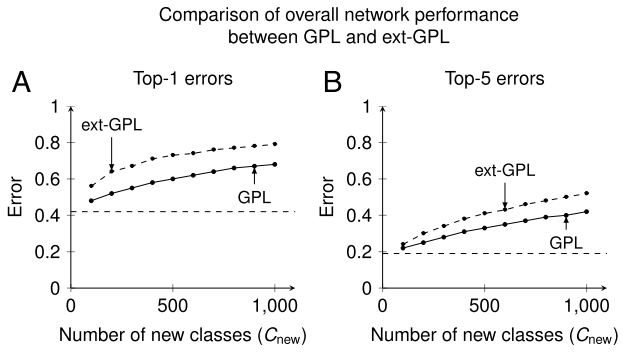

Fig. 17. $1 \leqslant C_{new} \leqslant 1000$ Development of the average network error as a function of the number of new classes added to the network for classes that are learned according to the GPL (Section 4) and ext-GPL (Section 5.2) methods.

that has been chosen as an examples for a poor classification performance. However, this improvement also comes with an increase of the overall error-rate; see Fig. 17.

This is reminiscent of the situation observed in Section 7.1, where the performance of the single category could be improved beyond the GPL result, but the overall performance declined (Fig. 15). Here, the analog of this overfitting may be due to a classification vector that is chosen too long. Actually, it may not be the total length $|\mathbf{v}_c|$ that was chosen too long as we assured that it has the average value of the old categories, $|\mathbf{v}_c| = \ell^{\pm}$, with $\ell^{\pm}$ defined in Eq. (15). It may rather be the relative contributions of the components given by $\bar{\mathbf{y}}^{\pm}$ and $\bar{\mathbf{x}}^{\pm}$ that could play a role here. With Eq. (23), these entered with equal weight. It may, however, be necessary to let them enter the new classifier with different weights.

For the case in Section 7.1, we concluded that an appropriate stopping criterion should be found to avoid overfitting. Here, with respect to ext-GPL some other criterion may be necessary which determines the appropriate normalizations and weights of the different layers that contribute to the classification vector as this acts on the classification and the pre-classification spaces. The finding of such optimization criteria is beyond the scope of the current work. It may, however, be the subject of future research directions.

8. Summary and discussion

Given a pre-learned Deep Convolution Network (CNN), we considered a simple method ("Global Prototype Learning", GPL) which constructs additional classifiers for new categories based on using only few training examples ("few-shot learning"). The resulting classifiers are prototypes which are derived from the activation that the training examples generate in the final global feature layer before classification. Correspondingly, we refer to this space as classification space. As a concrete example for the pre-learned starting point, we considered the CNN described in Chatfield et al. (2014). The core results may be summarized in the following six statements.

First, the method allows a quick classification procedure that succeeds to classify a substantial portion of the new categories with an error rate that is comparable to the performance of the old classifiers (the ones of the pre-learned CNN). While the old classifiers have error rates that are centered around an average value, the new classes have a rather broad error distribution where a large portion of the errors are of the order of or even below the values of the pre-learned classes.

Second, while training of the old classifiers of the pre-learned CNN required examples galore, the new classifiers result from a

few-shot learning. Only few examples suffice to reach a plateau in the performance. The classification does not decisively improve if the number of examples is increased beyond say ten examples, both for the "good" and the "bad" classifiers.

Third, we linked the performances of the classifiers to the data distribution in the classification space. This was done by applying the dimensional reduction method of t-SNE (Maaten & Hinton, 2008). The classification space is defined as the space on which the classifiers are acting, that is, the output of the final global feature layer before the classification layer. Using the t-SNE method, we found that good and bad classification performances are linked to separable and non-separable clustering in the classification space, respectively. In fact, this property is not restricted to the new classes but also explains bad performances of old classifiers.

Fourth, using some examples, we illustrated how the mixing of data clusters of different classes in the classification space corresponds to commonalities of the related input images.

Fifth, we also defined a backpropagation based variant of the learning procedure, here referred to as normalized backpropagation learning (NBL). We compared the error rates resulting from the GPL with those from NBL. Although the latter procedure needs more computational steps for the construction of the classifiers, we did not find a decisive improvement in error rates. Without introducing an early-stopping criterion , the overall learning performance with respect to all learned categories rather declined. In this context, it is crucial to note that in the course of the discussed work, we applied the restriction that no examples for the previously learned examples are allowed to be included into the learning process. The foregoing learning processes are assumed to be finished and all the examples for the learned categories have been consolidated in the corresponding previously learned classification vector.

Sixth, we also studied whether extending the classification space to the two final global feature layers allows for even better classification results for the new categories. We found no clear improvement and again related this result to the clustering as it shows up through using the t-SNE method, now applied to the extended classification space.

These observations, on one hand, illustrate once again the powerful generalization ability that a backpropagation learned CNN has beyond the classification of the categories that originally enter the learning process. In the context of the present work, this is confirmed by the observation that a substantially large number of new classifiers constructed with the GPL method show a performance that is comparable to the classification performance of classifiers of the original CNN. On the other hand, our discussion makes obvious that problems with classification performance (for new and old classes) may be deeply routed in the data that define the classes as these may be inherently difficult or even inappropriate for assuring a good classification. Take as an example the discussed walrus category where walruses may be understood as resembling sea lions with tusks. Considering a particular walrus example which showed the classification space data point to be positioned inside the sea lion data cloud, it turned out that the input image of this example shows three walruses from behind so that no tusks are visible (left image in Fig. 11). Obviously, this is a walrus example that can hardly be distinguished from a sea lion example. Or, consider the class of bagels. If images show the bagel from the side, thus hiding the characteristic opening in its center, it is hardly distinguishable from bread or sometime from cracker. And so we found the three categories bagels, bread (French loaf), and cracker to be mixed, making also the new classification of crackers difficult.

Such examples point to the fact that the road towards improving on the generalization performance of CNNs may require a higher degree of dynamical processing that is beyond current CNNs. Take, for example, the walrus versus sea lion case that we just mentioned. If an input shows an animal from behind so that it could represent a walrus or a sea lion then this uncertainty should be detected and not be overwritten by a false recognition. Instead of making a not sufficiently justified decision, the detected uncertainly may trigger emphasis on other aspects like comparing the size of the animal with respect to other objects that are recognized in the background. Or, with respect to the mentioned bagel classification, seeing an object from the side that could be a bread or a bagel may require to search for other hints or just to keep the classification uncertain until additional information arrived. Also, it points to the necessity of choosing the training examples in a most informative manner (also humans who learn a new category will not perform successfully if the training examples are misleading or do not point to the essential characteristics of the new class).

The dynamical character of a procedure that recognizes "undecidable" cases and uses these to trigger the search for other features or, at least, imply a different weighting of features, is a procedure that is yet outside of the reach of feedforward CNN classification. For new categories inside of this range, however, simple procedures to include new categories based on few examples may be quite successful for a remarkable portion of new categories.

Acknowledgment

It is a pleasure to thank Jochen Triesch for valuable discussions.

References

Akata, Z., Malinowski, M., Fritz, M., & Schiele, B. (2016). Multi-Cue Zero-Shot Learning with Strong Supervision, arXiv preprint arXiv:1603.08754.

Chatfield, K., Simonyan, K., Vedaldi, A., & Zisserman, A. (2014). Return of the devil in the details: Delving deep into convolutional nets. In *British machine vision conference*.

Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., & Fei-Fei, L. (2009). Imagenet: A large-scale hierarchical image database. In *IEEE conference on computer vision and pattern recognition, 2009*. CVPR 2009, (pp. 248–255). IEEE.

de Vries, H., Memisevic, R., & Courville, A. (2016). Deep learning vector quantization. In *European symposium on artificial neural networks, computational intelligence and machine learning*.

Donahue, J., Jia, Y., Vinyals, O., Hoffman, J., Zhang, N., & Tzeng, E. et al. (2014). DeCAF: A deep convolutional activation feature for generic visual recognition. In *ICML* (pp. 647–655).

Fink, M. (2005). Object classification from a single example utilizing class relevance metrics. *Advances in Neural Information Processing Systems*, *17*, 449–456.

Fritzke, B. (1995). A growing neural gas network learns topologies. *Advances in Neural Information Processing Systems*, *7*, 625–632.

Hariharan, B., & Girshick, R. (2016). Low-shot visual object recognition, arXiv preprint arXiv:1606.02819.

Hecht, T., & Gepperth, A. (2016). Computational advantages of deep prototype-based learning. In *International conference on artificial neural networks* (pp. 121–127). Springer.

Jetley, S., Romera-Paredes, B., Jayasumana, S., & Torr, P. (2015). Prototypical Priors: From Improving Classification to Zero-Shot Learning, arXiv preprint arXiv:1512.01192.

Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems* (pp. 1097–1105).

Lake, B. M., Salakhutdinov, R. R., & Tenenbaum, J. (2013). One-shot learning by inverting a compositional causal process. In *Advances in neural information processing systems* (pp. 2526–2534).

Lake, B. M., Salakhutdinov, R., & Tenenbaum, J. B. (2015). Human-level concept learning through probabilistic program induction. *Science*, *350*(6266), 1332–1338.

Lampert, C. H., Nickisch, H., & Harmeling, S. (2009). Learning to detect unseen object classes by between-class attribute transfer. In *IEEE conference on computer vision and pattern recognition, 2009*. CVPR 2009, (pp. 951–958). IEEE.

LeCun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning. *Nature*, *521*(7553), 436–444.

LeCun, Y., Bottou, L., Bengio, Y., & Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, *86*(11), 2278–2324.

Li, F.-F., Fergus, R., & Perona, P. (2006). One-shot learning of object categories. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *28*(4), 594–611.

Maaten, L. v. d., & Hinton, G. (2008). Visualizing data using t-SNE. *Journal of Machine Learning Research (JMLR)*, *9*(Nov), 2579–2605.

Miller, G. A. (1995). WordNet: a lexical database for English. *Communications of the ACM*, *38*(11), 39–41.

Pan, S. J., & Yang, Q. (2010). A survey on transfer learning. *IEEE Transactions on Knowledge and Data Engineering*, *22*(10), 1345–1359.

Rezende, D. J., Mohamed, S., Danihelka, I., Gregor, K., & Wierstra, D. (2016). One-Shot Generalization in Deep Generative Models, arXiv preprint arXiv:1603.05106.

Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., et al. (2015). Imagenet large scale visual recognition challenge. *International Journal of Computer Vision (IJCV)*, *115*(3), 211–252. http://dx.doi.org/10.1007/s11263-015-0816-y.

Sato, A., & Yamada, K. (1996). Generalized learning vector quantization. *Advances in Neural Information Processing Systems*, 423–429.

Schroff, F., Kalenichenko, D., & Philbin, J. (2015). Facenet: A unified embedding for face recognition and clustering. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 815–823).

Vedaldi, A., & Lenc, K. (2015). MatConvNet –convolutional neural networks for MATLAB. In *Proceeding of the ACM int. conf. on multimedia*.

Wang, Y., & Cottrell, G. W. (2015). Bikers are like tobacco shops, formal dressers are like suits: recognizing urban tribes with caffe. In *2015 IEEE winter conference on applications of computer vision* (pp. 876–883). IEEE.

Wang, Y.-X., & Hebert, M. (2016). Learning to learn: model regression networks for easy small sample learning. In *European conference on computer vision* (pp. 616–634). Springer.

Zhang, Z., & Saligrama, V. (2015). Zero-shot learning via semantic similarity embedding. In *Proceedings of the IEEE international conference on computer vision* (pp. 4166–4174).