Feature Contraction: New ConvNet Regularization in Image Classification

Vladimir Li
vlali@kth.se
Atsuto Maki
atsuto@kth.se

School of Electrical Engineering and Computer Science (EECS),
KTH Royal Institute of Technology
SE-100 44, Stockholm, Sweden

Abstract

This paper presents a low-cost and effective regularization for artificial neural networks. Based on the paradigm of deep convolutional networks (ConvNets) optimization, we introduce the *feature contraction* simply by computing the L2 loss of the feature vector from a certain layer, and adding it to the standard cross-entropy loss. The key idea is to alleviate the vanishing gradient by allowing higher cross-entropy loss while promoting a more informative softmax categorical distribution. We show in our experiments that the new regularization contributes to a significant performance increase in image classification tasks. Our approach can also be used with other methods of regularization such as dropout, and applicable to broader problem domains by its nature.

1 Introduction

In supervised learning the model complexity need be well balanced against the richness of the information contained in training data in order to prevent overfitting. This also holds for deep convolutional networks (ConvNets), which this paper is concerned with in particular in the context of image classification, and therefore some regularizers are often utilized, e.g. a weight decay [9] added to the loss function, dropout [16], and data augmentation [8].

The loss for optimizing neural network models is normally based on the cross-entropy between a *one-hot* representation (target class label) and a categorical distribution computed as an output of the network through K-way softmax (K: the number of classes). In this common setting of the cost function, the network is trained to make the categorical distribution as close as possible to the *hard target* whose elements are all zero except the one for the true class being '1'. It is widely accepted that minimizing the cross-entropy loss is an objective that reasonably conforms to the goal of well achieving the classification accuracy.

In this paper, however, we propose a new loss called *feature contraction* while addressing possible issues in the standard strategy of using the softmax cross-entropy loss on its own. First, the gradient of the loss may suffer more from *vanishing gradient* as the training proceeds; the softmax output values can saturate when the differences between input elements become extreme, but this is exactly the expected effect by minimizing the cross-entropy. In fact, from the derivative of the cross-entropy loss, it is derived that the gradient approaches zero as the softmax probability converges to the one-hot encoded ground truth vector.

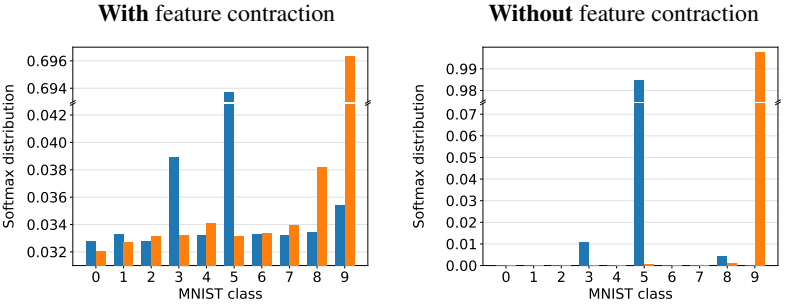

Figure 1: Effects of the feature contraction on softmax categorical distributions. Shown are averaged softmax outputs for samples from Class '5' (blue) and '9' (orange) in the MNIST dataset. Left: the feature contraction loss is added. Right: only the cross-entropy loss used. In left, in particular, samples of Class '5' appear to be more similar to Class '3' than to the other categories, likewise Class '9' to '8'.

Secondly, in the view of maximizing the information from an input image, we question if it is ideal for a model to be trained in a way to identify all categories except the correct one as simply *wrong*, even those similar to the correct class. This question is related to the insight underlying the *dark knowledge*: "The relative probabilities of incorrect answers tell us a lot" which Hinton et al. based for their use of "soft target" in knowledge distillation [6]. Now, one could instead let relative probabilities appear on the other input of cross-entropy; we may encourage the network to carry probabilities of incorrect classes, reflecting the closeness to the correct class better. For example, it would make sense to allow the network, given an input image of a horse, to answer for {Horse, Sheep, Pencil} as {0.6, 0.3, 0.1} (after softmax) rather than {0.96, 0.03, 0.01}. Thus, we argue that similarities between different classes count as rich information which can be exploited.

Based on those insights, we introduce the *feature contraction* as a simple but powerful approach to cope with the above-described concerns in the solo use of cross-entropy loss. We define a new loss as the L2 norm of a feature vector in the top layer(s) of ConvNets, to be directly added to the cross-entropy loss. The key idea is to enforce the discrete distributions in the feature vector to become less peaky for the use in softmax cross-entropy loss. That is, the new loss penalizes high values in the feature vector, whether positive or negative, while keeping the values on other units less suppressed. To the best of our knowledge this is the first regularizer that counteracts the cross-entropy loss in this explicit manner.

Figure 1 illustrates the effect of feature contraction, where the averages of the softmax distribution collected for the samples from Class '5' (blue) and '9' (orange) are displayed. It can be observed that with feature contraction (left) the probabilities for incorrect classes are less ignorable, e.g. the probability of '3' for '5', whereas without feature contraction (right) the probabilities are mostly suppressed in the orders of magnitude ranging from 10^{-2} to 10^{-12}. Feature contraction thereby brings about an effect of preserving the magnitude of the gradient, which helps to achieve better classification accuracy. We will show its benefit on a few standard image classification tasks while also comparing the results with other regularization methods such as weight decay and dropout. We performed the study on the handwritten digit dataset MNIST [10] as well as on two other datasets, Oxford-102 [12] and CUB200-2011 [13], in the setting of transfer learning. VGG-16 [15] and ResNet-50 [5] are used in our studies on those two datasets.

The contributions of this paper can be summarized as (i) a concept of allowing the elements of a softmax distribution to reflect the categorical closeness in image classification, (ii) *feature contraction* loss as a novel low-cost tool for regularizing artificial neural networks, and (iii) supporting evidence that shows improved classification accuracy.

2 Method

We present our new regularization, the *feature contraction*, preceded by the motivation that a more uniform logits distribution can improve the behavior of the gradient for network optimization. We also analyze the influence of the feature contraction loss in a network thereafter.

2.1 Motivation

In the common settings of ConvNets optimization for image classification problems, the cross-entropy loss is usually employed as a measure of error between two distributions: the ground truth $\mathbf{y} = \{y_i\}$ $(i = 1, ..., K)$ in one-hot encoding ($y_c = 1$ for the correct class c, $y_i = 0$ $\forall i \neq c$) and the categorical probability computed from unit activations, *i.e.* logits, of the output layer $\mathbf{z} = \{z_i\}$ $(i = 1, ..., K)$ in conjunction with the softmax function, $\sigma(z_i)$. That is,

$$\mathcal{L}_{xe}(\mathbf{y}, \mathbf{z}) = -\sum_{i=1}^{K} y_i \log \sigma(z_i) \qquad \text{where} \qquad \sigma(z_i) = \frac{e^{z_i}}{\sum_{k=1}^{K} e^{z_k}}. \tag{1}$$

Each element, $\sigma(z_i)$, is bounded in the range $[0, 1]$, and the sum of those is always one. The derivative of $\mathcal{L}_{xe}$ with respect to each logit z_i can then be beautifully derived [4] as

$$\frac{\partial \mathcal{L}_{xe}}{\partial z_i} = \sigma(z_i) - y_i, \tag{2}$$

and hence it is equal to the difference between the corresponding elements in the ground truth vector and the softmax output distribution. When those values become exactly the same for all the elements, the cross-entropy, $\mathcal{L}_{xe}$, takes its minimum, and therefore the gradient close to zero as seen in equation 2. In general, when using only the cross-entropy loss, the outputs of the softmax function are to converge to the corresponding ground truth vector, leading inevitably to a vanishing gradient. Thus, the goal of minimizing the loss contradicts the preference of maintaining some gradient.

Importantly, we argue that at the cost of allowing higher cross-entropy loss we could benefit to alleviate vanishing gradient at the later stages of network training; we could consider adding a constraint that promotes a less peaky (more uniform) softmax distribution such that the output would be kept from fully converging to the corresponding ground truth. This would give the network more room to carry on the search for a better parameter configuration.

The remaining question is how to encourage a more uniform distribution for the output of the softmax function. A straightforward solution would be to enforce the input to the softmax, i.e. logits, to be more uniform as it would result in a more uniformly distributed softmax output accordingly.

2.2 Feature Contraction

Based on the reasons given in section 2.1, we propose an additional loss term that penalizes high values among the logits and forces the logits distribution to be as uniform as possible. That is, we introduce the *feature contraction* loss as a regularization term to be used with cross-entropy loss in the cost function. To be concrete, we define it based on the L2 norm of the logits vector, typically coming from the last prediction layer, or the second last layer; the feature contraction loss is applied to the feature before the Rectified Linear Unit (ReLU) activation function as appropriate to avoid causing a skewed distribution (see section 2.3).

Denoting the values stored in a ConvNet layer before the ReLU activation function as $\tilde{\mathbf{z}}^{(l)}$ with $l \in \{1,...,N\}$ as the layer number and N the total number of layers in the network, we define the feature contraction loss as

$$\mathcal{L}_c = \eta \|\tilde{\mathbf{z}}^{(l)}\|_2^2, \tag{3}$$

and the cost function as a whole as

$$E = \mathcal{L}_{xe} + \mathcal{L}_c, \tag{4}$$

where η is the parameter that controls the contraction, namely how strongly large logits are penalized. For convenience, we use

$$\eta = \eta_c / 2 s_b \dim(\tilde{\mathbf{z}}^{(l)}), \tag{5}$$

where η_c that adjusts η in practice, the batch size is s_b, and $\dim(\tilde{\mathbf{z}}^{(l)})$ is the number of elements in $\tilde{\mathbf{z}}^{(l)}$. This reflects the fact that the preferred value of η varies depending on the batch size and the number of units in the layer to which the feature contraction is applied.

2.3 Analyses of the Feature Contraction Loss

In order to analyze the influence of the feature contraction loss, let us first consider layer l, which is fully connected with the previous layer. $\tilde{\mathbf{z}}^{(l)}$ will be called the feature vector in the sequel, and $\mathbf{z}^{(l)} = \max(0, \tilde{\mathbf{z}}^{(l)})$ indicates a vector after the ReLU activation. $\mathbf{z}^{(l-1)}$ is then the vector after the ReLU in layer $l-1$.

The effect of the feature contraction loss as suggested in equation 3 is to reduce the magnitude of feature $\tilde{\mathbf{z}}^{(l)}$. If we chose to use $\mathbf{z}^{(l)}$ after the ReLU activation as a feature to contract instead of $\tilde{\mathbf{z}}^{(l)}$, only the positive portion of the signal in $\tilde{\mathbf{z}}^{(l)}$ gets contracted whereas the negative part would remain unaffected. This would make the distribution of the elements in $\tilde{\mathbf{z}}^{(l)}$ skewed and biased toward negative values as a consequence. In our studies we noticed that feature contraction on $\tilde{\mathbf{z}}^{(l)}$ in equation 3 induced a negatively skewed distribution in feature vector $\tilde{\mathbf{z}}^{(l-1)}$ in the previous layer. This indicates that the effect of feature contraction may be propagating backward to the earlier layers in such a way as if we had applied feature contraction on vector $\mathbf{z}^{(l-1)}$. Corresponding analyses using an example are provided in section 4.

For another observation, the L2 norm of feature vector $\tilde{\mathbf{z}}^{(l)}$ can be decomposed as:

$$\|\tilde{\mathbf{z}}^{(l)}\|_2^2 = \|\mathbf{W}^{(l)}\mathbf{z}^{(l-1)} + \mathbf{b}^{(l)}\|_2^2 = \|\mathbf{W}^{(l)}\mathbf{z}^{(l-1)}\|_2^2 + 2(\mathbf{W}^{(l)}\mathbf{z}^{(l-1)})^T\mathbf{b}^{(l)} + \|\mathbf{b}^{(l)}\|_2^2 \tag{6}$$

where $\mathbf{W}^{(l)}$ and $\mathbf{b}^{(l)}$ are the weight matrix and bias vector of layer l, respectively. Feature contraction can be seen as a combination of three factors: the L2 norm of a feature vector

$\mathbf{z}^{(l-1)}$ transformed by $\mathbf{W}^{(l)}$, a cross term, and the L2 norm of the biases $\mathbf{b}^{(l)}$. In our empirical study (in section 3.1), the first term has by far the largest impact on the classification accuracy while the bias-related terms have a marginal effect. We can then derive the derivative of the first term w.r.t. $w_{ij}^{(l)}$ to understand how it affects the update of a single weight $w_{ij}^{(l)}$,

$$\frac{\partial \|\mathbf{W}^{(l)}\mathbf{z}^{(l-1)}\|_2^2}{\partial w_{ij}^{(l)}} = 2z_j^{(l-1)}\langle \mathbf{W}_i^{(l)}, \mathbf{z}^{(l-1)}\rangle \tag{7}$$

where $z_j^{(l-1)}$ is the j-th element in $\mathbf{z}^{(l-1)}$, $\mathbf{W}_i^{(l)}$ is the i-th row of $\mathbf{W}^{(l)}$, and $w_{ij}^{(l)}$ is an element in $\mathbf{W}^{(l)}$. From equation 7, it should be noticed that the update to parameter $w_{ij}^{(l)}$ is not only a function of itself as in the case of weight decay, but a function of input $\mathbf{z}^{(l-1)}$ as well as the corresponding row of $\mathbf{W}_i^{(l)}$.

3 Evaluations

We test feature contraction on three different datasets and three network architectures; we performed experiments on the digit dataset MNIST to examine the effect as well as on the Oxford-102 and CUB200-2011 datasets using the VGG-16 and ResNet-50 models. We applied the feature contraction first on the feature vectors from the last layer and then on those from the second last layer, where the results in our initial studies were more promising by the latter. Thus, we chose to relax the original definition of the feature contraction loss (section 2.2) and mainly apply it to the second last layer unless otherwise stated.

3.1 MNIST

The **MNIST** [10] dataset is a collection of handwritten digits. It consists of 60K training and 10K test images of dimension 28x28 pixels. We based our experiments on an off-the-shelf convolutional neural network available in a TenserFlow tutorial [11] (denoted "mnistNet" in the following). The mnistNet consists of two 5x5 convolutional layers each followed by a 2x2 max-pooling layer with stride two and two fully connected layers as shown in table 1. The feature contraction was applied after fc3 layer.

We did not use any image transformations for the data augmentation apart from scaling and shifting the image intensity. We applied a dropout, when we chose to use it, with a probability of 50% before the prediction layer. The experiments were performed with a mini-batch gradient descent, using a batch size of 100, learning rate of 10^{-2}, and momentum of 0.9.

Three sets of experiments were performed with this setup:

1. First, we investigated how the contraction weight η_c affects the accuracy by varying it between 10 and 10^{-4} while training the mnistNet for 20k iterations. Table 2 shows the test errors: all values of η_c improve the classification accuracy over the case of using only the cross-entropy loss while we also see the need to choose η_c properly.

2. Second, we compare the performance of the feature contraction with those by dropout and a combination of the two. The experiments ran for 100K steps, and the results at the end of the training are shown in table 3. In this experiment the feature contraction

Layer	conv1	pool1	conv2	pool2	fc3	fc4
Kernel Size, Stride	5,1	2,2	5,1	2,2	-	-
Number of filters or units	32	-	64	-	1024	10

Table 1: The configuration of the mnistNet. The convolutional and pooling kernels are symmetric with a size of $p \times p$ where p is the first number on Row 2 and a stride of $q \times q$ where q is the second number on Row 2. Max-pooling was used as a pooling operator and ReLU as the activation function.

Parameter (η_c)	10	1	0.1	0.01	0.001	0.0001	0
Test error	0.53%	**0.46%**	0.57%	0.62%	0.65%	0.69%	0.70%

Table 2: Test error for MNIST trained with different levels of feature contraction parameter. $\eta_c = 0$ corresponds to the case where no feature contraction is used. See equation 5 for η_c.

	Test error
(1) No regularization	0.70%
(2) Dropout	0.51%
(3) Feature contraction ($\eta_c = 1$)	**0.41%**
(4) Combination of (2) and (3)	0.46%

Table 3: Test error for mnistNet trained (100K steps) with different regularization methods.

achieves better improvement than dropout. One can observe that the combination of the two regularization techniques did not necessarily boost the performance.

3. Further, we tested the individual terms of the L2 norm defined in equation 6. We ran 100k iterations with each term as an additional loss. It turned out that the first term, $\|\mathbf{W}^{(l)}\mathbf{z}^{(l-1)}\|_2^2$, is the dominant factor in the feature contraction loss (test error **0.47%**) whereas no significant gains were observed by using either the second term (0.69%) or the third (0.67%). This was performed purely for the sake of analyzing the contributions to the feature contraction loss, but it shows that the feature contraction can be used as a whole, without such a division, also from the perspective of implementation.

3.2 Oxford-102 and CUB200-2011

To verify the capability of the feature contraction, we compare the effect of different regularization techniques on more challenging datasets. We fine-tune VGG-16 and ResNet-50 on two fine-grained tasks using the feature contraction and/or other means of regularization. We describe the datasets, networks, and methodologies and present the experimental results.

3.2.1 Datasets

The **Oxford-102** [17] dataset consists of 8189 images of flowers distributed over 102 classes. The training and validation set consists of 10 images per class, and the remaining is allocated to the test set. In our experiments we faithfully follow the original data split without mixing the training and validation data.

	VGG-16		ResNet-50	
Optimizer	SGD		RMSprop	
Batch size	64		32	
Learning rate pretraining	-		10^{-4}	
Learning rate fine-tuning	10^{-3}		10^{-5}	
Momentum	0.9		-	
RMSprop decay	-		0.9	
Weight decay	$5 \cdot 10^{-4}$		10^{-5}	
Contraction parameter (η_c)	10^{-2}		10^{-3}	
Dropout keep probability	0.5		0.5	
	Oxford-102	CUB200	Oxford-102	CUB200
Number of epochs pretraining	-	-	60	60
Number of epochs fine-tuning	1000	200	200	60

Table 4: Settings used for training of VGG-16 and ResNet-50. The feature contraction parameter (η_c) was chosen through some trials. Those parameters are not necessarily optimized for the specific tasks, and therefore the resulting performance may not be optimal.

The **CUB200-2011** [18] is a dataset of 200 different species of birds. It consists of 11788 images. The suggested split has 5994 images for training and 5794 images for testing. In our experiments, we randomly removed one sample of each class from the training set to create a validation set. Hence, our split consists of an equal amount of train and test samples.

For the training we augmented the data by random jittering, subtle scaling, and rotation. We resized the images to 250×250 and applied zero padding to keep the aspect ratio. During the training we applied random crop and random flip across the vertical axis. The validation and testing are performed on a single center crop.

3.2.2 Experiments

We used two different networks for our experiments: VGG-16 [15] and ResNet-50 [5]. The former has been studied well, has simple and well behaved training properties while the latter has a slightly more complex structure, much deeper with residual connections and batch normalization. In both architectures they have the same data preprocessing step making the experimental setup simpler while still having a good model diversity. The off-the-shelf models used in this study are available at [17].

The models were initialized with ImageNet [13] pretrained parameters. For both VGG-16 and ResNet-50, we replaced the last prediction layer with a layer that matches the number of classes according to the dataset. The parameters of that layer were initialized randomly by Xavier's initialization [6]. The rest of the parameters were loaded from the pretrained models. For the ResNet-50 the prediction layer was pretrained on each dataset, and the fully pretrained model was used as initialization for the fine-tuning experiments. During the fine-tuning of ResNet-50, we froze the batch normalization layers, so that they were not updated. In the case of VGG-16 we initiated fine-tuning experiments without pretraining the last layer. The implementation details for the training are summarized in table 4.

In the case of both network architectures five models were saved with even intervals. The test accuracy for the best model and particular training setup are presented in table 5. The only difference among the experiments are the extra regularizations while the rest of

	Oxford-102		CUB200-2011	
	VGG-16	ResNet	VGG-16	ResNet
(0) Pretrained prediction layer	-	86.55%	-	69.54%
(1) No regularization	83.09%	87.59%	75.04%	74.54%
(2) Weight decay	83.69%	86.13%	73.46%	74.85%
(3) Dropout	84.78%	89.32%	77.10%	77.39%
(4) Feature Contraction	**87.38%**	91.49%	78.55%	**82.57%**
(5) Combination of (3) and (4)	85.97%	92.49%	**79.74%**	81.08%
(6) Combination of (2),(3) and (4)	79.61%	**92.55%**	77.74%	80.10%
Reduction of error rate (1) $\rightarrow$ (4)	25.37%	31.43%	14.06%	31.54%
Reduction of error rate (3) $\rightarrow$ (4)	17.08%	20.32%	6.33%	23.91%

Table 5: Classification accuracy for the Oxford-102 and CUB200-2011 dataset.

R-CNN [2, 7]	Branson [1]	TLAN [19]	NAC [14]	DVAN [21]	Zhang [20]
69.0%	75.7%	77.9%	81.0%	79.0%	84.54%

Table 6: Classification accuracy on the CUB200-2011 dataset from some other work.

the settings were kept identical. In the case of VGG-16 we applied the feature contraction after fc7 while for ResNet-50 we added dropout and feature contraction after global pool, just before the prediction layer.

In the experiments it is observed that adding feature contraction always improves the classification accuracy. The resulting error rate reduction indicates that feature contraction has a relatively higher impact on ResNet-50 than on VGG-16 (see the last two lines in table 5). Moreover, combining Dropout and feature contraction can be beneficial in some cases, further improving upon the prediction accuracy.

In table 6, for reference, we list the classification accuracy on the CUB200-2011 which are achieved by other studies with different advanced techniques (we defer to the original sources for details). This shows how our simple addition of feature contraction performs comparatively well.

4 Discussion

Figure 2 summarizes the progression of the three training sessions of the mnistNet. The feature contraction was applied in the training in two ways: after the last layer (green) or after the second last layer (orange) while the blue curve represents the case without any regularization. The test performances for the runs with feature contraction are superior to the case with the cross-entropy loss only, while the cross-entropy loss is maintained higher by the virtue of feature contraction. This indicates that the feature contraction restricts the cross-entropy loss not to be minimized. Thanks to the higher cross-entropy loss, we manage to maintain larger and more stable gradients (rightmost in figure 2). Note that the gradient for the training session with feature contraction after the last layer (green) is a combination of the gradients propagating from the cross-entropy and the feature contraction losses. Figure 3 also shows the progression of the three sessions, but it does so in terms of the sparsity of features; we can observe that the feature contraction also encourages sparsity in the feature

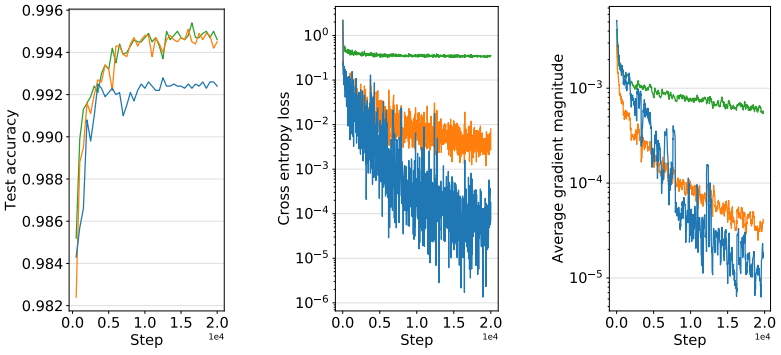

Figure 2: Analyses of training sessions of mnistNet. From left to right: the test accuracy, the cross-entropy loss, and the average gradient for the weights of the last layer are shown. Three sessions are color-coded: (i) using only the cross-entropy loss (blue), (ii) using the feature contraction on the second last layer (orange), and (iii) using the feature contraction on the prediction layer (green).

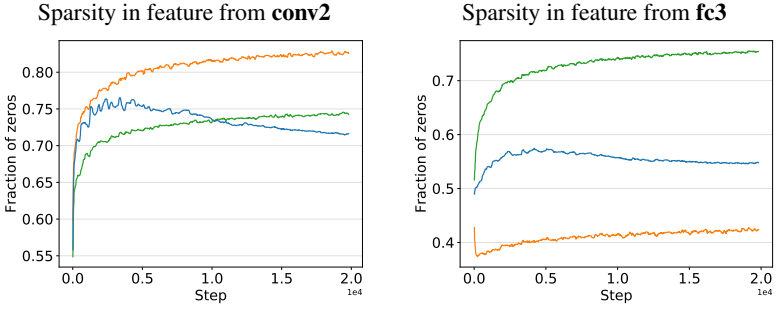

Figure 3: The feature sparsity measured by fraction of zeros in the same three training sessions as in figure 2 (common color-code). Left: the sparcity of the feature from conv2 layer. When the feature contraction is applied after fc3 layer (orange), the sparsity in the feature from the conv2 layer increases. Right: the sparcity of the feature from fc3 layer. The effect of the increased sparsity in the feature from the fc3 layer can also be observed for the training where the feature contraction is applied after fc4 (green).

from the preceding layer (see the caption of figure 3 for further explanations), which could be another reason for the improved performance.

We have presented some evidence of improved performance by the feature contraction on fine-grained classification tasks using deep ConvNets in the setting of transfer learning as well as using a shallow network. As seen in table 6, there exist several methods that rely on a mechanism of detecting the region of interest for the improved classification accuracy. It is among the future work to see if feature contraction can also boost those results. Given that feature contraction is beneficial on relatively small-scale datasets, it is interesting to investigate if the merit would also scale up to a large-scale dataset in conjunction with a deep network, e.g. VGG-16 trained on ImageNet. In addition, from the theoretical viewpoint, the feature contraction is not restricted to image classification tasks and can be applied to other tasks where softmax cross-entropy loss is employed.

5 Conclusion

We proposed the feature contraction as a simple and effective regularizer for training deep convolutional networks in image classification. It is a possible solution to alleviate the issue of vanishing gradient involved in the solo use of the cross-entropy loss; adding the new simple regularization loss allows more informative softmax distribution while better retaining the flow of the gradient, which improves the classification accuracy. Promising results have been shown in our early experiments using different convolutional networks for a few classification tasks in a transfer learning setting. Furthermore, there were no evident decreases in the training speed (iterations per time unit) due to added feature contraction loss. The feature contraction can also be utilized as a general method with other forms of regularizations or network architectures. Future work will be directed to see how it performs in other settings, i.e. large-scale datasets and other problem domains.

Acknowledgement The authors gratefully acknowledge the support by the Swedish Research Council for this research. We wish to thank Yang Zhong for useful discussions and NVIDIA Corporation for their generous donation of NVIDIA GPUs.

References

[1] Steve Branson, Grant Van Horn, Pietro Perona, and Serge J. Belongie. Improved bird species recognition using pose normalized deep convolutional nets. In *BMVC*. BMVA Press, 2014.

[2] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *CVPR*, pages 580–587, 2014.

[3] Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In *AISTATS*, volume 9 of *JMLR Proceedings*, pages 249–256. JMLR.org, 2010.

[4] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016.

[5] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, pages 770–778. IEEE Computer Society, 2016.

[6] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. *arXiv*, 1503.02531, 2015.

[7] Jonathan Krause, Hailin Jin, Jianchao Yang, and Fei-Fei Li. Fine-grained recognition without part annotations. In *CVPR*, pages 5546–5555, 2015.

[8] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. Imagenet classification with deep convolutional neural networks. In *NIPS*, pages 1106–1114, 2012.

[9] Anders Krogh and John A. Hertz. A simple weight decay can improve generalization. In *NIPS*, pages 950–957. 1992.

[10] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. In *Proceedings of the IEEE*, pages 2278–2324, 1998.

[11] Ingo Lütkebohle. A guide to tf layers: Building a convolutional neural network. https://www.tensorflow.org/tutorials/layers. [Online; accessed 23-April-2018].

[12] Maria-Elena Nilsback and Andrew Zisserman. Automated flower classification over a large number of classes. *Sixth Indian Conference on Computer Vision, Graphics & Image Processing*, pages 722–729, 2008.

[13] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, 115(3):211–252, 2015.

[14] Marcel Simon and Erik Rodner. Neural activation constellations: Unsupervised part model discovery with convolutional networks. In *ICCV*, pages 1143–1151. IEEE Computer Society, 2015.

[15] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *CoRR*, abs/1409.1556, 2014.

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

[17] Tensorflow. Tensorflow-slim. URL https://github.com/tensorflow/models/tree/master/research/slim.

[18] C. Wah, S. Branson, P. Welinder, P. Perona, and S. Belongie. The Caltech-UCSD Birds-200-2011 Dataset. Technical report, 2011.

[19] Tianjun Xiao, Yichong Xu, Kuiyuan Yang, Jiaxing Zhang, Yuxin Peng, and Zheng Zhang. The application of two-level attention models in deep convolutional neural network for fine-grained image classification. In *CVPR*, pages 842–850. IEEE Computer Society, 2015.

[20] Xiaopeng Zhang, Hongkai Xiong, Wengang Zhou, Weiyao Lin, and Qi Tian. Picking deep filter responses for fine-grained image recognition. In *CVPR*, pages 1134–1142. IEEE Computer Society, 2016.

[21] Bo Zhao, Xiao Wu, Jiashi Feng, Qiang Peng, and Shuicheng Yan. Diversified visual attention networks for fine-grained object classification. *IEEE Trans. Multimedia*, 19 (6):1245–1256, 2017.