ESANN 2020 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Online event, 2-4 October 2020, i6doc.com publ., ISBN 978-2-87587-074-2.
Available from http://www.i6doc.com/en/.

New Results on Sparse Autoencoders for Posture Classification and Segmentation

Doreen Jirak and Stefan Wermter

University of Hamburg - Knowledge Technology
Vogt-Kölln-Str. 30, 22527 Hamburg - Germany

Abstract. This paper is a sequel on posture recognition using sparse autoencoders. We conduct experiments on a posture dataset and show that shallow sparse autoencoders achieve even better performance compared to a convolutional neural network, state-of-the-art model for recognition tasks. Also, our results support robust image representation from the autoencoder model rendering further finetuning unnecessary. Finally, we suggest using sparse autoencoders for image segmentation.

1 Introduction

Command gestures or counting with fingers are popular examples of hand postures called *emblems*. Due to their significant impact in communication, the recognition of postures play a vital role in human-robot interaction (HRI). Deep learning models, especially convolutional neural networks (CNN), show high recognition accuracy for benchmarks but remain data-hungry and computationally demanding. Image augmentation is used to generate necessary data, however, this may not apply when particular properties have to be maintained, as e.g. in medical imaging. Finally, getting insights into the image representations help to understand the underlying learning principles. These aspects motivated a previous study [1] on sparse autoencoders, where we demonstrated their applicability for posture recognition and the influence of learned representations on performance. In this paper, we selected another posture dataset to find further evidence of our results. Additionally, we compare the results with a CNN. Furthermore, we give a proof of concept how sparse autoencoders can be used for image segmentation.

2 Dataset and Methodology

We used the NUS-I dataset [2], which comprises 10 posture classes performed by 24 subjects. We downsampled the images to pixel size 28×28 [1] and added a *no hand* condition computed from the mean background pixel value, yielding 264 images and 11 classes. As this work is a sequel to our previous study, we use sparse autoencoders to learn the postures. They are unsupervised neural networks trained by minimizing the mean-squared error between an original image x and the reconstructed image y based on a set of neuron weights W in a hidden layer representing image features. Additionally, a softmax classifier is used for the classification. A finetuning step can be applied to further optimize the features and, thus, the classification. The autoencoders can also be stacked,

ESANN 2020 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Online event, 2-4 October 2020, i6doc.com publ., ISBN 978-2-87587-074-2. Available from http://www.i6doc.com/en/.

which means that after training the first autoencoder, the resultant features are passed to a second autoencoder and so on. The objective function $\mathcal{J}$ for sparse autoencoder training is:

$$\mathcal{J}(x,y) = \min_{W} \Big[\sum_{i=1}^{N} (h_W(x^{(i)} - y^{(i)})^2 + \lambda(||W||_2^2) + \beta \sum_{j=1}^{m} KL(\rho||\hat{\rho}_j) \Big]$$

where the first sum term is the reconstruction error from the autoencoder for N images, λ is the regularization to prevent overfitting. KL is the Kullback-Leibler divergence between the desired firing rate ρ and the average $\hat{\rho}$ summed over m neuron activations in the hidden layer of the autoencoder. Parameter β penalizes the divergence. The role of ρ is crucial as it controls the sparsity of firing, i.e. the reaction of neurons to visual stimuli, to balance neural activity and energy costs. Here, this means that neurons specialize to encode different image features.

In contrast, a CNN consists of multiple filter maps of different sizes with initally random weights which are trained using backpropagation along a layer hierarchy. The cascading system of convolutions and pooling operations is responsible why CNNs can learn composite shapes like objects or faces, which make them a state-of-the-art model for recognition tasks. However, training the weights to represent significant image content is highly data-dependent and the question is, therefore, how reliable CNNs perform for small datasets and whether sparse autoencoders can be alternative models for limited-size data. To answer this question, we set up CNNs and evaluate their performance for three popular optimizers: stochastic gradient descent with the momentum term γ (*sgdm*), root-mean-square propagation (*rmsprop*) [3] providing a adaptive learning rate, and the adaptive moment estimation [4] (*adam*) providing adaptive learning and a term similar to training with momentum. In brief, let the network parameters be denoted by weights $\omega \in \mathbb{R}^d$ and the gradient Δ of an objective function $J(\omega)$. An update over the parameter space using *sgdm* over time t is computed as:

$$v_t = \gamma v_{t-1} + \eta \Delta_\omega J(\omega; x^{(i:i+n)}; y^{(i:i+n)})$$
$$\omega = \omega - v_t$$

where γ is the momentum term, η is the learning rate, and training is done with mini-batches of size $(i:n)$ for input-output pairs $(x;y)$. The parameter update for *rmsprop* [3] is :

$$\omega_{t+1} = \omega_t - \frac{\eta}{\sqrt{E[g^2]_t + \epsilon}} g_t$$

where $E[g^2]_t$ denotes the average of squared gradients g at time t, g_t is the gradients at time t, and ϵ is a small value to prevent division by zero. The *adam* optimization [4] integrates first and second moment estimates, denoted as $\hat{m}$ and $\hat{v}$ into the gradient updates:

$$\omega_{t+1} = \omega_t - \frac{\eta}{\sqrt{\hat{v}_t} + \epsilon} \hat{m}_t$$

ESANN 2020 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Online event, 2-4 October 2020, i6doc.com publ., ISBN 978-2-87587-074-2. Available from http://www.i6doc.com/en/.

where $\hat{m}_t$ and $\hat{v}_t$ are computed as:

$$\hat{m_t} = \frac{m_t}{1-\phi_1^t}$$
$$m_t = \phi_1 m_{t-1} + (1-\phi_1) g_t$$

$$\hat{v_t} = \frac{v_t}{1-\phi_2^t}$$
$$v_t = \phi_2 v_{t-1} + (1-\phi_2) g_t^2$$

3 Experimental Settings

We use 70% of the images for training and 30% for test [1], the logistic sigmoid activation function and backpropagation with scaled conjugate gradient optimization (scg) [5]. We ran 20 trials with 500 epochs for each layer L_1 and L_2 with numbers of neurons $N_1 = 100$ and $N_2 = 50$. Table 1 summarizes all parameters. To compare with [2], we also used an image size of 128×128[1] and a split of 50% training and 50% test (in reference paper: N=2 [2]). Training time was around 10 minutes (CPU, 3.3GHz).

λ_{L_1}	β_{L_1}	ρ_{L_1}	λ_{L2}	β_{L_2}	ρ_{L_2}	N_1	N_2
0.001-0.01	1-4	0.1-0.4	0.01	1-4	0.1-0.4	100	50

Table 1: Autoencoder parameter

We used a standard CNN for comparison reason and to check whether they achieve better results. To compare with [2], we kept the image resolution 120×160. We used a 3-layer CNN as we observed no reasonable performance with less layers, certainly due to CNNs relying on computing image compositions. Table 3 summarizes the filter sizes and number of maps in layers $\mathcal{L}$. We used a stride $s = 1$, if not specified differently. As a trade-off regarding computing times, we discarded the pooling operation in $\mathcal{L}_3$. No zero padding to the image was necessary. We used the ReLU activation function at each layer and batch normalization. We fixed $\lambda = 1e-8$ and the number of epochs to 100 with a mini-batch size of 32. At every epoch, those sets were randomly shuffled. All other settings follow the protocol for autoencoders. The training time for all optimizers was around $1.5h$ (CPU, 3.3GHz). For *sgdm*, we set the momentum term be $\gamma = 0.9$ and the learning rate $\eta = 0.01$ [3]. We set up experiments with either a constant or an adaptive learning rate η, the latter multiplied by 0.1 at every 10th epoch. We set the learning rate $\eta = 0.001$ for *adam* and *rmsprop*. We let $\phi_1 = 0.9$, $\phi_2 = 0.999$, and $\epsilon = 1e-8$ [4].

[1]The input to the autoencoder is a square matrix

ESANN 2020 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Online event, 2-4 October 2020, i6doc.com publ., ISBN 978-2-87587-074-2. Available from http://www.i6doc.com/en/.

	$\mathcal{L}_1$	$\mathcal{L}_2$	$\mathcal{L}_3$
conv	10×10 (10)	4×4 (10)	2×2 (10)
pool	5×5 (5)	2×2, $s = 2$	-

Table 2: Configuration of the CNN.

4 Results and Evaluation

We evaluate the test set regarding parameters ρ and β as they influence the performance. Table 3 shows the median and mean accuracy averaged over 20 trials for both for classification with the autoencoder and softmax classifier only, and after finetuning (cf. [1]). For $\rho = 0.1$ and $\rho = 0.2$ the performance was

Accuracy (%)	AE+softmax		Finetuning	
ρ	mean	median	mean	median
0.1	84.75	84.81	85.44	86.08
0.2	85.00	84.81	85.50	86.08
0.3	83.99	84.18	84.30	84.81
0.4	82.91	82.28	83.41	83.54

Table 3: Average accuracy with fixed $\beta = 1$ for one layer sparse autoencoder (L_1).

equal, so we conclude that for small dataset ρ it suffices to provide a low value. Increasing β did not lead to performance improvement. Trial analyses using the McNemar test with significance level $\alpha = 0.01$ revealed no statistically significant performance differences between autoencoder only and finetuning. Using the L_2 design yield no significant performance improvements: we obtained an accuracy of 87.34% for $\rho_1 = \rho_2 = 0.1$ and $\beta_1 = 4$, $\beta_2 = 1$ when finetuning. This is marginally different compared to L_1 and across all other parameter configurations. Using a larger image size to compare with [2], we obtained 84.81% accuracy but notably trained on only 50% of the images. The McNemar test showed no statistical difference between the autoencoder and the finetuning. With a computation time of ≈ 5 hours (CPU, 3.3GHz), dropping finetuning for time compensation becomes a crucial factor. Table 4 shows the results from different optimization schemes including dropout. The table suggests a performance gain for dropout=0.5, however, we observed strong fluctuations when training with *rmsprop*. That no dropout seems the best option can possibly be explained by the limited size of the dataset where generalization is less an issue.

5 Learning Representations for Image Segmentation

As this work is a sequel to a previous study, we revisit the Triesch dataset [6] (JTD) where we showed the impact of image backgrounds (white, black,

ESANN 2020 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Online event, 2-4 October 2020, i6doc.com publ., ISBN 978-2-87587-074-2. Available from http://www.i6doc.com/en/.

%	adam	rmsprop	sgdm
no dropout	**80.26**	76.32	73.68
dropout 0.1	77.63	75.00	72.37
dropout 0.5	78.95	79.61	76.32

Table 4: Median accuracy for the NUS-I dataset for different optimizers.

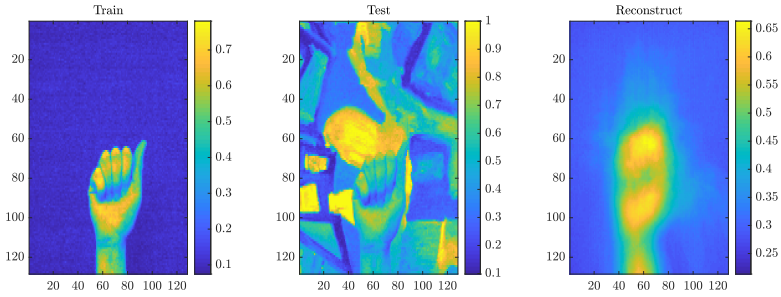

Fig. 1: Results from the segmentation.

complex) on the performance using the autoencoder architecture. Especially the complex patterns are challenging and postures are even hard to identify with the human eye. Originally developed to apply hand graphs, we hypothesized to use the learned representations of the sparse autoencoders for image segmentation [1]. We show a proof of concept of this idea. A naive approach would be to just multiply the black images with the complex ones, as "black" equals to a 0 in an image. We observed that is indeed not the case and this operation does not lead to the desired segmentation. Alternatively, using the encodings[2] from trained sparse autoencoder on black images can be applied (see Figure 1) as it learns significant hand features like fingertips. For final segmentation, we binarized all images at pixel value 0.5 (normalized) and computed the Jaccard coefficient. Figure 2 shows that our approach is superior, for 141 images we obtained a higher Jaccard coefficient using the encodings compared to the multiplication.

6 Conclusion

This sequel to our previous study supports that sparse autoencoders are viable models for posture recognition, further supported by a comparison with a standard CNN, which did not show superior performance. The advantage using sparse autoencoders is that they are fast to train without provision of large, labeled datasets. Although we did not improve the original performance, we obtained good performance for a shallow sparse autoencoder with a small image size and low number of encoding neurons ($\approx 1/7$ compression), which indicates that this model is able to learn a sufficient image representation. Our present

[2]We used the penalty $\beta = 10$

ESANN 2020 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Online event, 2-4 October 2020, i6doc.com publ., ISBN 978-2-87587-074-2.
Available from http://www.i6doc.com/en/.

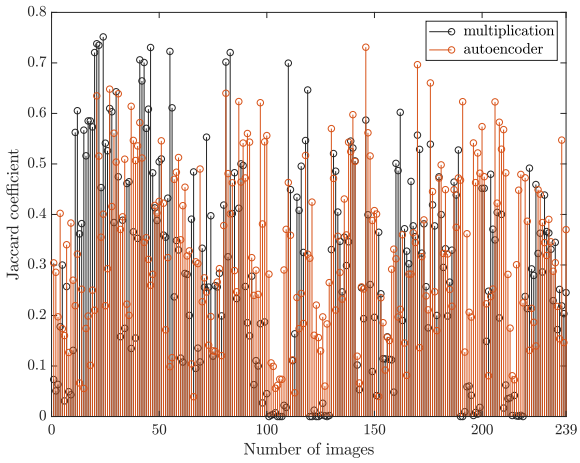

Fig. 2: The Jaccard coefficients for the segmentation.

study is not limited to postures but provides interesting insights into computational models for application constrained by limited data such as medical images, which prohibits image augmentation to preserve characteristics of images from, e.g., brain tissue. We sketched an approach how to use encodings for image segmentation and we are currently running segmentation experiments using "hands in the wild" to extrapolate our study to realistic environments.

References

[1] Doreen Jirak and Stefan Wermter. Sparse autoencoders for posture recognition. In *Proceedings of the International Joint Conference on Neural Networks (IJCNN 2018)*, Jul 2018.

[2] P. PramodKumar, Prahlad Vadakkepat, and Ai Poh Loh. Hand posture and face recognition using a fuzzy-rough approach. *I. J. Humanoid Robotics*, 7:331–356, 2010.

[3] Tijmen Tieleman and Geoffrey Hinton. Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude, 2012.

[4] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980, 2014.

[5] Martin Fodslette Møller. A scaled conjugate gradient algorithm for fast supervised learning. *Neural Networks*, 6(4):525 – 533, 1993.

[6] Jochen Triesch and Christoph von der Malsburg. Classification of hand postures against complex backgrounds using elastic graph matching. *Image and Vision Computing*, 20(13):937 – 943, 2002.