Unsupervised Feature Extraction – A CNN-Based Approach

Daniel J. Trosten[1(✉)] and Puneet Sharma[2]

[1] UiT Machine Learning Group, Department of Physics and Technology, UiT The Arctic University of Norway, Tromsø, Norway
daniel.j.trosten@uit.no
[2] Department of Engineering and Safety (IIS-IVT), UiT The Arctic University of Norway, Tromsø, Norway

Abstract. Working with large quantities of digital images can often lead to prohibitive computational challenges due to their massive number of pixels and high dimensionality. The extraction of compressed vectorial representations from images is therefore a task of vital importance in the field of computer vision. In this paper, we propose a new architecture for extracting such features from images in an unsupervised manner, which is based on convolutional neural networks. The model is referred to as the Unsupervised Convolutional Siamese Network (UCSN), and is trained to embed a set of images in a vector space, such that local distance structure in the space of images is approximately preserved. We compare the UCSN to several classical methods by using the extracted features as input to a classification system. Our results indicate that the UCSN produces vectorial representations that are suitable for classification purposes.

1 Introduction

An inherent property of digital images is their large, and often varying dimensionality. A typical image can contain thousands, or even millions of pixels, meaning that the curse of dimensionality [2] can be devastating if not handled properly. This fact has led to the development of a multitude of techniques for computing compressed representations of images, that attempt to retain important information in the representation. Moreover, it is often required that for a database of images, the pairwise similarities between images are approximately preserved through the extraction process. This is vital for the performance of image classification systems, which rely on the quantitative viewpoint provided by a collection of representations. In general, it is easier to design and train a classification system which uses less complex decision functions (e.g. linear), as opposed to systems that rely on more complex decision functions. Hence, a feature extraction technique which is suitable for classification should transform an image database such that the categories one wishes to distinguish form compact and linearly separable "clusters" in the feature space.

Many of the widely used feature extraction techniques are based on the idea of one or more *image descriptors*, that can be computed either over the whole

M. Felsberg et al. (Eds.): SCIA 2019, LNCS 11482, pp. 197–208, 2019.
https://doi.org/10.1007/978-3-030-20205-7_17

image (globally), or from certain subregions of the image (locally). An example of a global descriptor is the *Histogram of Oriented Gradients* (HOG), which has been successfully applied to human recognition and tracking [8]. Examples of local descriptors include the *Scale Invariant Feature Transform* (SIFT) [15], *Speeded Up Robust Features* (SURF) [1], and *Gradient Location and Orientation Histogram* (GLOH) [17].

With these local descriptors, the main idea is to identify points of interest in the input image, and then compute vectorial descriptors based on neighborhoods around these points. The SIFT algorithm identifies keypoints by locating minima or maxima in a scale space, which is constructed by convolving the input image with discretized Laplacian of Gaussians at different scales. The SURF algorithm is based on a similar approach, in which the scale space is constructed using box-filters of varying widths, instead of Gaussians.

Once a set of local descriptors has been computed for an image, these have to be further quantized into a single vector representing the entire image. This can be achieved using the *Bag of Visual-words* (BOV) [7], in which all descriptors from the entire dataset are clustered with k-means, thus creating a vocabulary of visual-words. The feature vector for a given image is then the histogram produced when assigning each of its descriptors to one of the visual-words. Another approach to quantization is the *Improved Fisher Vector* (IFV) [11], which fits a Gaussian mixture model to the complete set of descriptors. Then, for a given image, the feature vector is constructed based on the average gradient of the mixture's log-likelihood, evaluated at each of the descriptors extracted from the image.

Recently, the image processing community has seen a shift in methodology with the introduction of models based on deep learning. The *convolutional neural network* (CNN) [14] has received much attention due to advancements in training strategies and computational capacity [13]. Although a vast majority of the CNN-based models are applied to supervised problems, there are no direct restrictions within the CNN architecture on performing unsupervised tasks. In other words, a modification in the training regime can be adopted to train a CNN-based model for feature extraction in an unsupervised manner.

In an unsupervised feature-extracting CNN, the learned feature vector – and therefore also its quality with respect to the task at hand will depend on the large number of parameters contained in the network. The development of training strategies, and more specifically, loss functions, for training deep learning architectures in an unsupervised manner is a field of research which still is in its early stages. Notable contributions within this field include loss functions designed for joint feature learning and clustering [26,27], as well as the *Convolutional Autoencoder* (CAE) [16].

The CAE consists of two CNNs, referred to as the encoder and decoder, respectively. The task of the encoder is to embed the input image in a vector space, while the task of the decoder is to reconstruct the input image based only on the embedded representation. These networks are trained together to minimize the mean squared reconstruction error, causing the encoder to learn a mapping which aims to preserve as much information about the input image as possible. When the training process terminates, the decoder is discarded and the encoder can be used as a feature extractor.

In this article, we propose the *Unsupervised Convolutional Siamese Network* (UCSN). It is a new deep learning based feature extractor which consists of a convolutional neural network that is trained end-to-end to learn an adaptive neighborhood embedding in an unsupervised manner, similarly to the weakly supervised *Siamese Networks* [6]. In contrast to the convolutional autoencoder [16], our model does not rely on a decoder network during training.

The rest of the paper is structured as follows: In Sect. 2, we provide an in-depth explanation of the proposed architecture. In Sect. 3, we explain the details of our experiments. In Sects. 4 and 5 we present and discuss our results, respectively. Finally, we provide some concluding remarks in Sect. 6.

2 Method

In this section we introduce the concept of siamese networks, and provide a detailed explanation of our proposed Unsupervised Convolutional Siamese Network.

2.1 Siamese Networks

Feature extraction can be regarded as learning a mapping from some input space X to an output space Y:

$$f_{\boldsymbol{\theta}} : X \to Y \subseteq \mathbb{R}^D$$

where f denotes the parameterized mapping, and $\boldsymbol{\theta}$ is a vector of parameters. D denotes the dimensionality of the extracted features, and depends on the design of the function f. In a *siamese network* [6,10,20], f is parameterized by either a fully-connected neural network or by a convolutional neural network. In both these cases, the final layer is responsible for producing the extracted feature. The chosen network is trained to minimize the contrastive loss function:

$$\mathcal{L}_c = \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} \left(a_{ij} ||\boldsymbol{y}_i - \boldsymbol{y}_j||^2 + (1 - a_{ij}) \max(0, c - ||\boldsymbol{y}_i - \boldsymbol{y}_j||)^2 \right), \quad (1)$$

where c is a hyperparameter, $||\cdot||$ denotes the Euclidean norm on Y, $\boldsymbol{y}_i = f_{\boldsymbol{\theta}}(\boldsymbol{x}_i)$ is the feature extracted from the input observation $\boldsymbol{x}_i$, and a_{ij} is an indicator variable defined as

$$a_{ij} = \begin{cases} 1, & (i,j) \text{ is a positive pair} \\ 0, & (i,j) \text{ is a negative pair} \end{cases}.$$

Two observations constitute a positive pair if they belong to the same class or category. Minimizing the contrastive loss function in Eq. (1) causes the network to learn a discriminative embedding by minimizing the distance within embedded positive pairs, while ensuring that the distance within negative pairs is sufficiently large. Thus, the network learns to embed points from the same class close to each other, while ensuring a large distance between points from different classes. The dependency on the a_{ij} in the loss function means that a siamese network requires weakly labeled data in the training process, making it a supervised model.

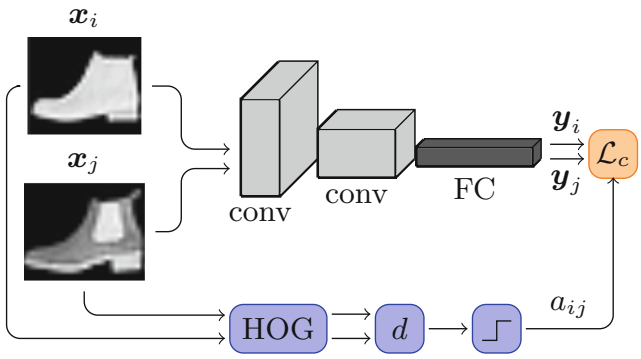


Fig. 1. Outline of the proposed architecture. The loss function considers distance structure in both the input space and the output space, ensuring that similar images are mapped to similar points, and vice versa. For more details on the exact CNN architecture, please see Table 4, Appendix A.

2.2 The Unsupervised Convolutional Siamese Network

To eliminate the need for weakly labeled data, we propose to construct positive and negative pairs based on a distance function in the input space. An overview of the architecture is provided in Fig. 1. The CNN receives as input a set of n images $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n \in X$, and produces the learned features $\boldsymbol{y}_1, \ldots, \boldsymbol{y}_n \in Y$. The model is trained by minimizing the loss function in Eq. (1), however the a_{ij} are now constructed in an *unsupervised* manner.

More specifically, the a_{ij} are constructed by labeling a pair as positive if the distance between them is less than ε, and negative otherwise:

$$a_{ij} = \begin{cases} 1, & d(\boldsymbol{x}_i, \boldsymbol{x}_j) < \varepsilon \\ 0, & d(\boldsymbol{x}_i, \boldsymbol{x}_j) \geq \varepsilon \end{cases},$$

where d is a suitable symmetric distance function on X, and ε was chosen such that the probability of two random training points being marked as a positive pair was approximately 0.02. This choice resulted in well-separated embeddings.

Note that this modification was previously done in [21], but they only considered the pixel-wise Euclidean distance. To construct a more robust distance function, we compute the Euclidean distance between HOG descriptors extracted from the respective images:

$$d(\boldsymbol{x}_i, \boldsymbol{x}_j) = ||\mathrm{HOG}(\boldsymbol{x}_i) - \mathrm{HOG}(\boldsymbol{x}_j)||.$$

3 Experiment Setup

3.1 Classifier

To quantitatively assess the performance of the extracted features with respect to classification, we train a linear support vector machine (SVM) [3,22] to classify

the learned features. The linear SVM was chosen as it is a linear classifier which is fast to train, and has been successfully applied to many classification tasks [23], making it a suitable classification component in the benchmarking process. We use a one-vs-rest SVM from the `scikit-learn` Python module, with default hyperparameters.

3.2 Models

In our experiments we evaluate two different versions of the UCSN. UCSN-Euc uses the Euclidean distance to determine positive and negative pairs, while UCSN-HOG uses the distance between HOG-features. We compare the proposed models to several state-of-the-art methods for image feature extraction [5]:

- *SIFT-BOV*: SIFT descriptors encoded with the bag of visual words (BOV) encoding, with a vocabulary size of 4096 visual words. The 128 dimensional descriptors were transformed to 80 dimensional vectors using PCA, before running k-means. The dimensionality reduction has been empirically found to improve the classification accuracy in similar models [4]. Each of the transformed descriptors were augmented with the normalized position of their respective keypoints, i.e. $(x/W, y/H)$, where (x, y) is the position of the keypoint in the image, and (W, H) is the width and height of the image, respectively.
- *SIFT-IFV*: SIFT descriptors encoded using the improved Fisher vector (IFV) encoding with 256 mixture components. Similarly to SIFT-BOV, the descriptors were reduced in dimensionality and augmented before fitting the Gaussian mixture model.
- *SURF-BOV*: Same as SIFT-BOV, but with SURF descriptors instead. Note that the 128 dimensional extended SURF descriptor was used. Dimensionality reduction with PCA, and position augmentation was done before running k-means.
- *SURF-IFV*: Same as SIFT-IFV, but with the 128 dimensional extended SURF descriptor. Dimensionality reduction with PCA, and position augmentation was done before fitting the Gaussian mixture model.
- *HOG*: The HOG descriptor.
- *CAE*: A fully convolutional autoencoder as described in [16]. Note that we train the model end-to-end instead of layer-wise, and use standard upsampling layers in the decoder, instead of the pooling scheme suggested in [16]. These modifications were made to reduce the computational complexity associated with training the model.

The dimensionality of the extracted features for the models can be found in Table 1, and the complete list of hyperparameter values for all models can be found in Table 4, Appendix A.

Note that the dimensionality of the extracted features is very large for some of the models and datasets. For the models based on local descriptors, the dimensionality can be adjusted by tuning either the number of clusters in BOV, or the number of mixture components in IFV. However, as pointed out in [4], values lower than 4096 clusters, or 256 mixture components, can lead to a drop performance.

Table 1. Dimensionalities of the extracted features. The last column indicates whether the dimensionality is a hyperparameter in the model.

Model	Dataset			Parameter
	SVHN [18]	Fashion-MNIST [25]	Cats and dogs [9]	
HOG	1764	1296	142884	✘
SIFT-BOV	4094	4096	4096	✔
SIFT-IFV	41984	41984	41984	✔
SURF-BOV	4094	4096	4096	✔
SURF-IFV	41984	41984	41984	✔
CAE	64	64	64	✔
UCSN-Euc/HOG	64	64	64	✔

Table 2. Properties of the different datasets used for evaluation.

	SVHN [18]	Fashion-MNIST [25]	Cats and dogs [9]
Image size	32×32	28×28	256×256
Color	RGB	Grayscale	RGB
# Samples	99289	60000	8192
# Classes	10	10	2
Contents	Natural images of house numbers	Clothing items	Natural images of cats and dogs

3.3 Datasets

The datasets chosen for evaluation are listed in Table 2. These constitute a range of object recognition tasks, without introducing auxiliary challenges, like multiple labels, large amounts of noise, and very high resolutions. Each dataset was randomly split into a training set, validation set, and test-set. 80 % of the images were used for training, 10 % were used for hyperparameter validation, while the last 10 % were used for performance evaluation. Note that the training set was used to train both the feature extractors and the corresponding SVMs.

3.4 Implementation

The experiments were implemented using the Python programming language, with the `NumPy` and `SciPy` modules for numerical computations. The HOG, SIFT and SURF descriptors were computed using the `OpenCV` module, while BOV and IFV were implemented on top of the `KMCuda` and `scikit-learn` modules, respectively. The SVM from the `scikit-learn` module was used, while the UCSN and the CAE were implemented and trained using the `TensorFlow` framework. Both these models were trained for 200 iterations, using stochastic mini-batches of size 1024 and the Adam optimizer [12].

Table 3. Classification accuracies for the different models and datasets.

Model	Dataset		
	SVHN	Fashion-MNIST	Cats and dogs
HOG	0.77	0.89	0.70
SIFT-BOV	0.50	0.74	0.69
SIFT-IFV	0.60	0.82	0.68
SURF-BOV	0.69	0.75	0.69
SURF-IFV	0.60	0.70	0.71
CAE	0.66	0.86	0.62
UCSN-Euc	0.38	0.80	0.59
UCSN-HOG	0.78	0.86	0.62

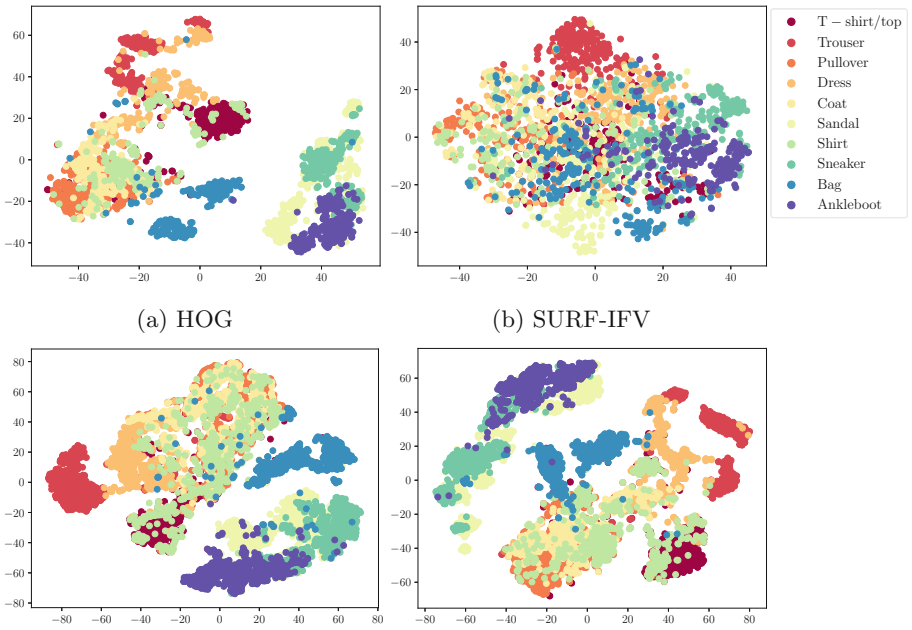
4 Results

The classification accuracies obtained from the different models and datasets are shown in Table 3. These indicate that the performance of the UCSN-HOG is comparable to the benchmark models. For the SVHN dataset, we observe that both UCSN-HOG and HOG outperform the rest of the models. This indicates that the HOG descriptors are suitable for discriminating between classes, and that the UCSN-HOG is able to exploit this information when extracting its features. This is not observed with the UCSN-Euc model, which performs significantly worse than all other models.

For the Fashion-MNIST dataset, UCSN-Euc is not as far behind, but still performs worse than UCSN-HOG, HOG, and the CAE. This again implies that these three models are able to extract features with better discriminative properties. On the other hand, the results for the Cats and Dogs dataset show that all deep learning based models perform worse than the other benchmark methods. Interestingly, this is opposite to what is commonly observed for fully supervised models, where CNN-based classifiers regularly outperform the classical approaches [5,13].

To further evaluate the validity of our results, the Fashion-MNIST features were projected to two dimensions using t-SNE [24] for selected models. Figure 2 shows the transformed features. These plots reveal a large difference in class separability, with the SURF-IFV features being significantly less separable than the other three. The classes are more separated by the CAE's representation, however, the features provided by HOG and UCSN-HOG still show better class separability. There is also a noticeable similarity between the HOG features and the UCSN-HOG features, which is expected since the UCSN-HOG is trained to preserve local HOG distances.

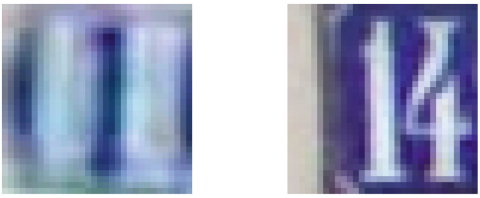
Figure 3 shows image pairs extracted from the SVHN and Cats and Dogs datasets, respectively. The normalized distances in the tables to the right are computed as $d_{\text{norm}} = \frac{d}{d_{\max}}$ where d is the distance between feature vectors extracted from the two images for the given model, and $d_{\max}$ is the maximum observed distance between any two extracted feature in the dataset, for the given model.



(a) HOG (b) SURF-IFV

(c) CAE (d) UCSN-HOG

Fig. 2. Plots of the features extracted using the HOG, SURF-IVF, CAE, and UCSN-HOG feature extractors. Note the large difference in class separability. The features were transformed to two dimensions using t-SNE [24].



HOG	SIFT-BOV
0.53	0.40
SIFT-IFV	SURF-BOV
0.69	0.71
SURF-IFV	CAE
0.79	0.48
UCSN-Euc	UCSN-HOG
0.47	0.18

(a)

HOG	SIFT-BOV
0.90	0.86
SIFT-IFV	SURF-BOV
0.85	0.40
SURF-IFV	CAE
0.71	0.62
UCSN-Euc	UCSN-HOG
0.58	0.91

(b)

Fig. 3. Image-pairs and their corresponding normalized distances. The distances are normalized by dividing the distance between extracted features, with the maximum observed distance, for each model and dataset.

The pair in Fig. 3a shows two images that are both labeled as "1". Note that the distance between UCSN-HOG embeddings is small, even though the image backgrounds are quite dissimilar, and the rightmost image contains an additional "4" which acts as a distraction. This indicates that UCSN-HOG has learned to recognize the "1" in the images. However, in Fig. 3b the cats depicted in the images are quite similar, but the distance between UCSN-HOG embeddings is large. This shows that UCSN-HOG is unable to pick up the similarities between these two images. This is also the case for most of the benchmark models, with the exception of UCSN-Euc and SURF-BOV.

5 Discussion

Although the CNN-based models, i.e. CAE and our proposed UCSN, perform well on the SVHN and Fashion-MNIST datasets, this was not observed with the Cats and Dogs dataset, where both these models performed worse than the other benchmark methods. This indicates that, when trained in an unsupervised manner, the filters learned by the CAE and the UCSN do not produce sufficiently discriminative representations in the output space. It is possible that this problem can be alleviated by employing different initialization techniques, for instance, through transfer learning [19], where the CNN is pre-trained as a classifier on a different dataset where ground truth labels are available. After pre-training, the network can be fine-tuned to the task at hand using the loss function in Eq. (1). Another option is to consider different distance functions in the image space, as this governs the local similarity structure of the extracted features. We emphasize that the proposed architecture can work with any symmetric distance function, making it applicable to a diverse set of domains. The exploration of these potential modifications is left as future work.

6 Conclusions

In this article, we outline a new architecture for image feature extraction that is based on convolutional neural networks. The proposed Unsupervised Convolutional Siamese Network (UCSN) is trained to embed a set of images in a vector space such that local distance structure is approximately preserved. We compare the UCSN model to several classical models by using the extracted features as input to a classification system. The overall results obtained from three different datasets indicate that the UCSN produces vectorial representations that are suitable for classification purposes. In the end, we suggest possible techniques that can be used for improving the performance of UCSN for feature extraction applications.

A Hyperparameters

Table 4. A summary of all the hyperparameters in all the models. Please consult the `OpenCV` documentation for an explanation of the HOG, SIFT and SURF hyperparameters.

Model	Parameter	Dataset		
		SVHN	Fashion-MNIST	Cats and Dogs
HOG	`blockSize`	8	8	8
	`blockStride`	4	4	4
	`cellSize`	4	4	4
	`nbins`	9	9	9
SIFT-BOV	`nfeatures`	64	64	32
	`sigma`	0.8	0.8	1.6
	`nOctaveLayers`	3	3	5
	`contrastThreshold`	0.001	0.001	0.004
	`k`	4096	4096	4096
SIFT-IFV	`nfeatures`	64	64	32
	`sigma`	0.8	0.8	1.6
	`nOctaveLayers`	3	3	5
	`contrastThreshold`	0.001	0.001	0.004
	`k`	256	256	256
SURF-BOV	`hessianThreshold`	0.0	0.0	200
	`nOctaves`	5	5	4
	`nOctaveLayers`	5	5	3
	`extended`	True	True	True
	`k`	4096	4096	4096
SURF-IFV	`hessianThreshold`	0.0	0.0	200
	`nOctaves`	5	5	4
	`nOctaveLayers`	5	5	3
	`extended`	True	True	True
	`k`	256	256	256
CAE	`layers`	Conv $5 \times 5 \times 32$ ReLU MaxPool 2×2 Conv $5 \times 5 \times 32$ ReLU MaxPool 2×2 Dense 64	Conv $5 \times 5 \times 32$ ReLU MaxPool 2×2 Conv $5 \times 5 \times 32$ ReLU MaxPool 2×2 Dense 64	MaxPool 2×2 Conv $7 \times 7 \times 16$ ReLU MaxPool 2×2 Conv $5 \times 5 \times 8$ ReLU MaxPool 2×2 Dense 64
UCSN	`layers`	Conv $5 \times 5 \times 32$ ReLU MaxPool 2×2 Conv $5 \times 5 \times 32$ ReLU MaxPool 2×2 Dense 64	Conv $5 \times 5 \times 32$ ReLU MaxPool 2×2 Conv $5 \times 5 \times 32$ ReLU MaxPool 2×2 Dense 64	MaxPool 2×2 Conv $7 \times 7 \times 16$ ReLU MaxPool 2×2 Conv $5 \times 5 \times 8$ ReLU MaxPool 2×2 Dense 64
	c	1	1	1
	ε (Euc/HOG)	12.41/5.10	5.86/3.39	110.56/48.1

References

1. Bay, H., Tuytelaars, T., Van Gool, L.: SURF: speeded up robust features. In: Leonardis, A., Bischof, H., Pinz, A. (eds.) ECCV 2006. LNCS, vol. 3951, pp. 404–417. Springer, Heidelberg (2006). https://doi.org/10.1007/11744023_32
2. Bellman, R.: Dynamic Programming. Princeton University Press, Princeton (1957)
3. Chang, C.C., Lin, C.J.: LIBSVM: a library for support vector machines. ACM Trans. Intel. Syst. Technol. **2**(3), 1–27 (2011). https://doi.org/10.1145/1961189.1961199
4. Chatfield, K., Lempitsky, V., Vedaldi, A., Zisserman, A.: The devil is in the details: an evaluation of recent feature encoding methods. In: Procedings of the British Machine Vision Conference 2011, pp. 76.1–76.12. British Machine Vision Association, Dundee (2011). https://doi.org/10.5244/C.25.76
5. Chatfield, K., Simonyan, K., Vedaldi, A., Zisserman, A.: Return of the devil in the details: delving deep into convolutional nets. arXiv:1405.3531 [cs], May 2014
6. Chopra, S., Hadsell, R., LeCun, Y.: Learning a similarity metric discriminatively, with application to face verification. In: 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, CVPR 2005, vol. 1, pp. 539–546, June 2005. https://doi.org/10.1109/CVPR.2005.202
7. Csurka, G., Dance, C.R., Fan, L., Willamowski, J., Bray, C.: Visual categorization with bags of keypoints. In: In Workshop on Statistical Learning in Computer Vision, ECCV, pp. 1–22 (2004)
8. Dalal, N., Triggs, B.: Histograms of oriented gradients for human detection. In: 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2005), vol. 1, pp. 886–893. IEEE, San Diego (2005). https://doi.org/10.1109/CVPR.2005.177
9. Elson, J., Douceur, J.J., Howell, J., Saul, J.: Asirra: A CAPTCHA that exploits interest-aligned manual image categorization. In: Proceedings of 14th ACM Conference on Computer and Communications Security (CCS). Association for Computing Machinery, Inc., October 2007
10. Hadsell, R., Chopra, S., LeCun, Y.: Dimensionality reduction by learning an invariant mapping. In: 2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition - Volume 2 (CVPR 2006), vol. 2, pp. 1735–1742. IEEE, New York (2006). https://doi.org/10.1109/CVPR.2006.100
11. Perronnin, F., Sánchez, J., Mensink, T.: Improving the Fisher Kernel for large-scale image classification. In: Daniilidis, K., Maragos, P., Paragios, N. (eds.) ECCV 2010. LNCS, vol. 6314, pp. 143–156. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-15561-1_11
12. Kingma, D.P., Ba, J.: Adam: a method for stochastic optimization. In: International Conference on Learning Representations (ICLR) (2015)
13. Krizhevsky, A., Sutskever, I., Hinton, G.E.: ImageNet classification with deep convolutional neural networks. Adv. Neural Inf. Process. Syst. **25**, 1097–1105 (2012)
14. Lecun, Y.: Generalization and network design strategies. In: Pfeifer, R., Schreter, Z., Fogelman, F., Steels, L. (eds.) Connectionism in perspective. Elsevier (1989)
15. Lowe, D.G.: Distinctive image features from scale-invariant keypoints. Int. J. Comput. Vis. **60**(2), 91–110 (2004). https://doi.org/10.1023/B:VISI.0000029664.99615.94
16. Masci, J., Meier, U., Cireşan, D., Schmidhuber, J.: Stacked convolutional auto-encoders for hierarchical feature extraction. In: Honkela, T., Duch, W., Girolami, M., Kaski, S. (eds.) ICANN 2011. LNCS, vol. 6791, pp. 52–59. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-21735-7_7

17. Mikolajczyk, K., Schmid, C.: A performance evaluation of local descriptors. IEEE Trans. Pattern Anal. Mach. Intell. **27**(10), 16 (2005)
18. Netzer, Y., Wang, T., Coates, A., Bissacco, A., Wu, B., Ng, A.Y.: Reading digits in natural images with unsupervised feature learning. In: Neural Information Processing Systems (NIPS), p. 9 (2011)
19. Pratt, L.Y., Mostow, J., Kamm, C.A.: Direct transfer of learned information among neural networks. In: Proceedings of the Ninth National Conference on Artificial Intelligence, AAAI 1991, vol. 2, pp. 584–589. AAAI Press, Anaheim (1991)
20. Shaham, U., Lederman, R.R.: Learning by coincidence: Siamese networks and common variable learning. Pattern Recogn. **74**, 52–63 (2018). https://doi.org/10.1016/j.patcog.2017.09.015
21. Shaham, U., Stanton, K., Li, H., Basri, R., Nadler, B., Kluger, Y.: SpectralNet: spectral clustering using deep neural networks. In: International Conference on Learning Representations (2018)
22. Smola, A.J., Schölkopf, B.: A Tutorial on Support Vector Regression (2004)
23. Theodoridis, S., Koutroumbas, K.: Pattern Recognition, 4th edn. Elsevier Academic Press, Amsterdam (2009). oCLC: 550588366
24. van der Maaten, L., Hinton, G.: Visualizing data using t-SNE. J. Mach. Learn. Res. **9**, 2579–2605 (2008)
25. Xiao, H., Rasul, K., Vollgraf, R.: Fashion-MNIST: A Novel Image Dataset for Benchmarking Machine Learning Algorithms, August 2017
26. Xie, J., Girshick, R., Farhadi, A.: Unsupervised deep embedding for clustering analysis. In: Proceedings of the 33rd International Conference on Machine Learning. Proceedings of Machine Learning Research, vol. 48, pp. 478–487, June 2016
27. Yang, J., Parikh, D., Batra, D.: Joint unsupervised learning of deep representations and image clusters. In: 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 5147–5156, June 2016. https://doi.org/10.1109/CVPR.2016.556