Learning Convolutional Neural Networks From Few Samples

Raimar Wagner, Markus Thom, Roland Schweiger, Günther Palm, and Albrecht Rothermel

Abstract— **Learning Convolutional Neural Networks (CNN) is commonly carried out by plain supervised gradient descent. With sufficient training data, this leads to very competitive results for visual recognition tasks when starting from a random initialization. When the amount of labeled data is limited, CNNs reveal their strong dependence on large amounts of training data. However, recent results have shown that a well chosen optimization starting point can be beneficial for convergence to a good generalizing minimum. This starting point was mostly found using unsupervised feature learning techniques such as sparse coding or transfer learning from related recognition tasks. In this work, we compare these two approaches against a simple patch based initialization scheme and a random initialization of the weights. We show that pre-training helps to train CNNs from few samples and that the correct choice of the initialization scheme can push the network's performance by up to 41% compared to random initialization.**

I. Introduction

For more than a decade, the Convolutional Neural Network (CNN) [1] has been among the most competitive neural network architectures for visual object recognition tasks. The concepts behind CNNs are biologically motivated and exploit intrinsic properties of natural images. Supplied with enough data, CNNs are highly competitive on a large variety of classification problems, even without unsupervised pre-training. CNNs only suffer when labeled data is scarce. Unfortunately, gathering large datasets with manually labeled data is both work-intensive and expensive. Besides adding architectural finesses such as rectifier neurons or contrast normalization [2] or adding additional distorted samples [3], training CNNs from few training examples was essentially addressed with two approaches prior to actual supervised parameter tuning: One could either transfer knowledge from other related or even unrelated tasks, or one could carry out an unsupervised pre-training stage to put the network in a beneficial state for the given problem.

In this work, we address the question of which learning strategy is beneficial when labeled data is scarce: unsupervised pre-training or transfer-learning. Furthermore, we evaluate if simple initialization heuristics can lead to a similar performance without the effort of unsupervised pre-training or transfer learning.

Raimar Wagner and Albrecht Rothermel are with the driveU / Institute of Microelectronics, University of Ulm, Ulm, Germany. (email: {raimar.wagner, albrecht.rothermel}@uni-ulm.de)

Markus Thom is with the driveU / Institute of Measurement, Control and Microtechnology, University of Ulm, Ulm, Germany. (email: markus.thom@uni-ulm.de)

Günther Palm is with the Institute of Neural Information Processing, University of Ulm, Ulm, Germany. (email: guenther.palm@uni-ulm.de)

Roland Schweiger is with Daimler AG, Ulm, Germany. (email: roland.schweiger@daimler.com)

II. Related Work

Learning CNNs from few samples is realized in the literature as either a sort of transfer learning or as a layer-wise unsupervised pre-training. In this work, we neglect approaches wherein a larger training dataset is narrowed down to reduce the training effort, for example by clustering the training dataset before training [4].

By transferring knowledge from different tasks, neural networks have shown to be able to learn new classes faster and from less training data [5]. Further, knowledge transfer is superior to reducing the net capacity when dealing with small training datasets [6]. Recently, it has been shown that transferring knowledge between different handwriting tasks is especially helpful when labeled data is scarce [7]. Instead of transferring knowledge from a domain specific task, generic pattern matching tasks (so-called pseudo tasks) generated from unlabeled data have been used to improve generalization performance on small datasets [8].

Unsupervised pre-training has shown astonishing results on various neural network architectures and datasets. It places the network in a beneficial state for a supervised refinement and guides the learning process towards a minimum, that provides better generalization capabilities compared to minima achieved from random starting points [9]. While most of the work deals with stacked auto-encoders [10], [11] and Deep Belief Nets [12], [13], only few approaches were made to pre-train CNNs. Ranzato et al. [14] have shown that unsupervised pre-training with a sparsity enforcing auto-encoder consistently improves classification performance on small datasets. Similarly, but with the use of a convolutional denoising auto-encoder, a CNN initialized with pre-trained filters has been shown to converge to a better minimum when refined with few samples [15].

III. Convolutional Neural Networks

In contrast to hand-crafted feature/classifier combinations which are widely used for vision tasks, CNNs form an end-to-end trainable hybrid feature-extraction/classification architecture. Inspired by the concept of simple and complex cells in the visual cortex [16], CNNs consist of alternating layers of feature extraction (simple cells) and subsampling (complex cells). In contrast to fully connected neural networks, the extensive use of shared weights reduces the number of degrees of freedom without the loss of expressive power, which makes CNNs easy to train with plain gradient descent. CNNs have been shown to be very competitive for visual recognition tasks [17], [18], [2], even outperforming human performance [19] on some datasets.

	Kernel	#Kernels	Feature Maps
Input			32×32 pixels
Filter Bank	7×7 pixels	128 (49)	26×26 pixels
Pooling	5×5 pixels	128 (49)	6×6 pixels
Fully Connected	5/10 neurons		

TABLE I: Network parameters of the used CNN. The CNN initialized from PCA has only 49 feature maps.

A. Layers

As CNNs are hierarchically structured as a stack of layers, this section briefly introduces the layer types used in this work. We distinguish three layer types while not accounting for the input layer which only holds the input pattern. The input of a specific layer is denoted with x, while y stands for the layer's output.

a) Convolutional Layer: In analogy to the simple cells of the visual cortex, the convolutional layer extracts local features by a convolution with a learned filter kernel. Every pair of input matrix x_i and filter kernel k_j is connected to a single output feature map $y_{i,j}$:

$$y_{i,j} = f\left(b_j + \sum\nolimits_i k_j * x_i\right).$$

Here, $*$ denotes the 2D convolution. With a size of $w_x \times h_x$ for the input matrix x_i and $w_k \times h_k$ for the filter kernel k_j, the feature map $y_{i,j}$ has the size $w_x - w_k - 1 \times h_x - h_k - 1$, which corresponds to a valid convolution border handling. The trainable weights in this layer are the filter kernel k_j and a single bias b_j per filter. As recommended by [20], the squashing function f is set to a scaled hyperbolic tangent, that is $f(x) = 1.7159 \cdot \tanh(2/3 \cdot x)$.

b) Pooling Layer: The pooling layers are equivalent to the complex cells of visual cortex. They are used to obtain a representation that is invariant against small translations and distortions. This is achieved by combining feature responses in non-overlapping spatial neighborhoods into one feature value. The original CNN utilized an average pooling [1] in every input region whereby all feature responses are uniformly weighted and averaged. For an input feature map x_i and an output feature map y_i, this corresponds to the mean operator in every non-overlapping region of size $r \times r$:

$$y_i = \underset{r \times r}{\operatorname{mean}}(x_i).$$

Variants include non-overlapping neighborhoods [17] and maximum operations [21]. Since this work is focused on weight initialization, we stick to the original average pooling operation.

c) Fully-Connected Layer: To form a classification decision, the spatial order from the pooling layer's feature responses is removed, and the feature responses are fed to a one-layered fully connected neural network with a weight matrix W and a bias vector b:

$$y = \operatorname{softmax}(b + Wx).$$

The output of the softmax transfer function represents the posterior distribution over the class labels.

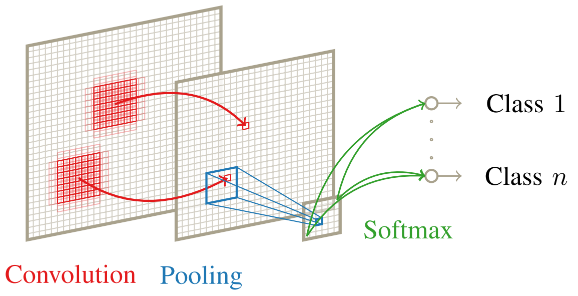


Fig. 1: The Convolutional Neural Network architecture used in this work: A convolutional feature extraction layer is followed by an average pooling layer and a fully-connected softmax output layer.

B. Hierarchy Construction

To form a hierarchical structure, the aforementioned layers are combined to an end-to-end trainable feature-extractor/classifier network. For the ease of simplicity we used an architecture with one layer of feature extraction, holding 128 filter kernels (49 for the PCA initialization) of size 7×7 pixels. This convolutional layer is followed by an average pooling of size 5×5 pixels. The pooled feature responses are then forwarded to a fully connected output layer with softmax neurons, which only computes the classification decision of the CNN. The network architecture is summarized in Table I and visualized in Fig. 1.

IV. Datasets

We have used two distinct datasets to evaluate our learning strategies, namely the 10-class "MNIST" database of hand-written digits [22] and the 5-class object recognition dataset "Small NORB" [18]. The full MNIST dataset consists of 60 000 samples for learning and 10 000 samples for testing. Besides padding each sample with 2 pixels on each side and a mean/variance normalization, no further pre-processing was applied. The Small NORB dataset consists of 24 300 samples each for learning and testing. Although it was designed as a stereo vision dataset, we only used the left image of each sample and grasped it as a monocular object recognition task. Similarly to [23], the whole dataset was downsampled to 32×32 pixels and mean/variance normalized in the pre-processing stage. Since after pre-processing both datasets had the same sample dimensionality, the same architecture could be used for both recognition tasks. Exemplary training samples from both datasets are depicted in Fig. 2.

To evaluate the proposed learning strategies, each CNN is trained either on a subset of the aforementioned datasets or the whole dataset. The subsets are constructed by randomly drawing a fixed number of samples per class from the entire dataset. These datasets are then frozen for the rest of the training procedure. The number of drawn samples per class and the resulting total sample count for each used dataset is shown in Table II.

(a) Small NORB

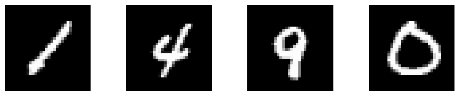

(b) MNIST

Fig. 2: Training samples from the Small NORB (a) and MNIST (b) dataset after preprocessing, each 32×32 pixels.

# Samples per class	MNIST # Samples	MNIST %	Small NORB # Samples	Small NORB %
30	300	0.5	150	0.6
100	1000	1.6	500	2.0
200	2000	3.3	1000	4.1
500	5000	8.3	2500	10.2
1000	10 000	16.6	5000	20.5
2000	20 000	33.3	10 000	41.1
3000	30 000	50.0	15 000	61.7
all	60 000	100.0	24 300	100.0

TABLE II: Subsets for learning using the MNIST dataset (10 classes) and the Small NORB dataset (5 classes).

V. Learning Strategies

In this work we evaluate five different learning strategies against each other under the aspect of the presence of scarce training data. Three approaches are of a very simple nature, namely drawing the initial kernel weights from a random distribution, initializing the kernels from randomly sampled patches from the training data or using the principal components of randomly sampled patches from the training data as kernel weights. Furthermore, we used filter kernels from a classifier trained on an unrelated dataset as a starting point for the supervised training. Finally, we evaluate the influence of pre-training with a sparse auto-encoder on the supervised CNN training.

A. Random Initialization

This initialization scheme is the most common for CNNs and works well when the amount of training data is sufficiently large. We used the recommendation of [20] for the random initialization of the weights with respect to the chosen squashing function, that is the entries of the filter kernels are sampled from a uniform distribution with zero mean and standard deviation set to the reciprocal of the square root of the neuron fan-ins.

B. Random Patches

The perhaps most simple unsupervised initialization method is drawing random samples from the learning dataset, and then extracting patches at random positions to initialize the filter kernels. This approach is similar to a common method for the initialization of Radial Basis Function networks [24], [25] and has previously been shown to be quite effective for discriminative feature extraction [26]. The computational complexity of this initialization scheme is only marginal, and due to its simplicity it is used as the baseline unsupervised method in the experiments.

C. Principal Component Analysis

Principal Component Analysis (PCA) is a standard tool for dimension reduction when working with high-dimensional data [27]. It orthogonally transforms observations which consist of a set of correlated variables into a set of linear uncorrelated variables, the so called principal components. The components are constructed successively such that each component explains most of the variability in the data while being orthogonal to the previous components. When applying PCA on image patches, the components range from low-frequency edge detectors in the first components to high-frequency components without a meaningful spatial structure in the other components [28]. We computed the PCA on random training dataset image patches of size 7×7 pixels and initialized a CNN with 49 features using these components.

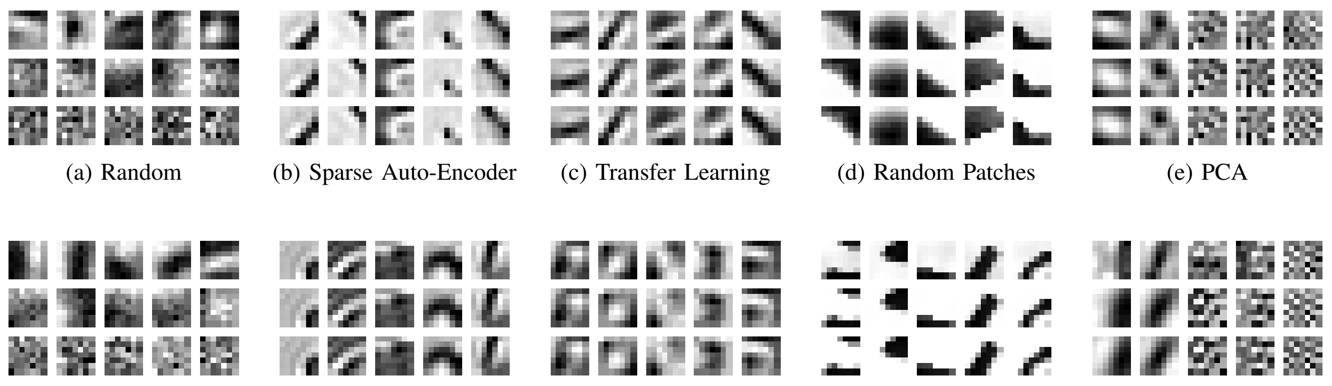
D. Transfer Learning

Transfer learning is inspired by the human learning behavior. We learn new tasks easier by using our previously acquired knowledge. Machine learning algorithms may also benefit from such previously learned information as was shown in Thrun's seminal work [5] for some neural network architectures. This seems natural for related datasets, like different handwriting tasks, where such an approach has shown to be very effective [7]. For image recognition tasks, common visual patterns like edges are even shared between unrelated datasets. This makes a knowledge transfer between such datasets promising [29], even though they are not related in a semantic sense.

Since the knowledge of a neural network lies in the weights that were trained on a specific task, the transfer is usually realized as a transfer of weights. It is, however, unclear whether transfer learning for CNNs can also help in the situation of unrelated datasets. We applied this scheme to transfer knowledge from MNIST to Small NORB and vice versa. The starting point of each transfer learning training is a classifier trained on the other respective dataset. Since both datasets differ in the number of classes, the output layer of the source network is capped and replaced by a randomly initialized output layer that has the appropriate number of output neurons. From this starting point, the whole network is refined in a supervised manner by gradient descent.

E. Sparse Auto-Encoder

Auto-encoder networks belong to the more sophisticated methods for unsupervised pre-training. They aim at transforming data points into an internal representation, such that the original data can be reconstructed from the internal representation without significant loss of information. There is a large variety of different approaches, ranging from linear ones that learn principal components [30], [31] to multi-layered, non-linear approaches [32]. In this work, we



(a) Random (b) Sparse Auto-Encoder (c) Transfer Learning (d) Random Patches (e) PCA

(f) Random (g) Sparse Auto-Encoder (h) Transfer Learning (i) Random Patches (j) PCA

Fig. 3: Features learned on Small NORB (a) – (e) and on MNIST (f) – (j). The figure shows the feature initialization (bottom row) and the refined feature after training with 100 samples per class (middle row) and all samples (top row).

used a single-layer, non-linear auto-encoder enriched with an explicit sparsity regularization as proposed by [33]. Here, the additional condition that the internal representations be sparsely populated forces their distribution to be evenly spaced in feature space. This special arrangement makes the distribution robust against failure of individual neurons. The usage of sparseness was inspired by biological neuronal networks that process visual stimuli, where this phenomenon was first observed [34], [35]. This is in fact a concept of efficiency, as only a very small fraction of the neuronal population can be active at any one time due to limited energy resources [36], [37].

The sparse auto-encoder we used for unsupervised initialization of the filter kernels can be considered a variant of matrix factorization [33]. Here, a matrix X of random patches extracted from the learning set should be factorized into a matrix of weights W and a matrix of internal representations H, such that $X \approx WH$. Further, given the input data the internal representations should be able to be inferred. For this, a bias vector b is introduced such that the internal representations can be computed through a one-layer neural network, in other words $H \approx f\left(b + W^T X\right)$ should hold. The vector b is here replicated in horizontal direction for the element-wise matrix addition. The squashing function f is set to a scaled, exponentiated hyperbolic tangent, namely $f(x) = 1.7 \cdot (\tanh(1.1 \cdot x))^3$. Raising $\tanh$ to the third power attenuates activation in the vicinity of the origin, such that sparseness is enhanced. The scaling of the exponentiated function was determined such that it is as close as possible to the squashing function defined in Sect. III-A in the Euclidean sense. The two approximations described above are then simultaneously optimized under the additional condition, that every row of H must fulfill a certain sparseness degree. As each row of H contains the activity of an individual feature over the entire learning set, this is equivalent to forcing a feature to be active on only a limited amount of training patches. For this, the normalized sparseness measure of [38] was used, and optimization was carried out using a projected gradient descent algorithm as also proposed by [38].

VI. Experiments

To evaluate which learning strategy is most helpful under the presence of scarce training data, we utilized the following experimental setup:

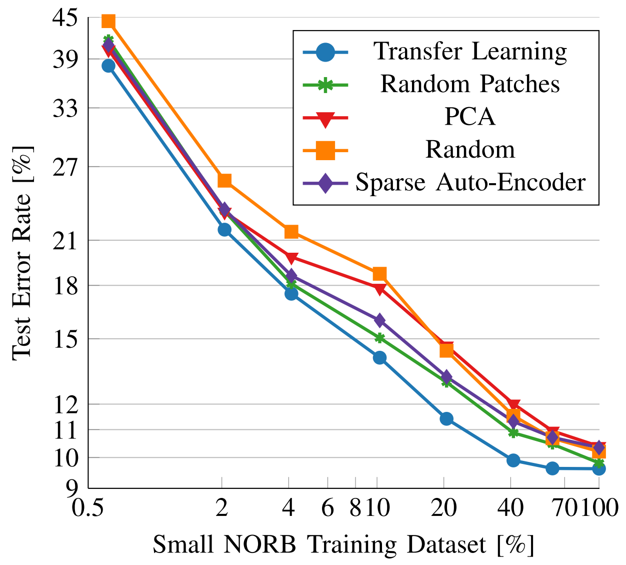
1) Initialization of the weights with one of the procedures from Sect. V on the basis of the picked data subset.
2) Refinement of the CNN in a supervised fashion with stochastic gradient descent until convergence.
3) The test error is reported based on the entire test dataset.

To account for random effects during initialization and supervised refinement, we conducted every experiment ten times. The learning rate for stochastic gradient descent was determined by grid search on 5000 samples for training and 5000 samples for validation disjointly drawn from the learning dataset. All experiments were conducted on basis of a publicly available GPU implementation (available under `http://code.google.com/p/cuda-convnet/`).

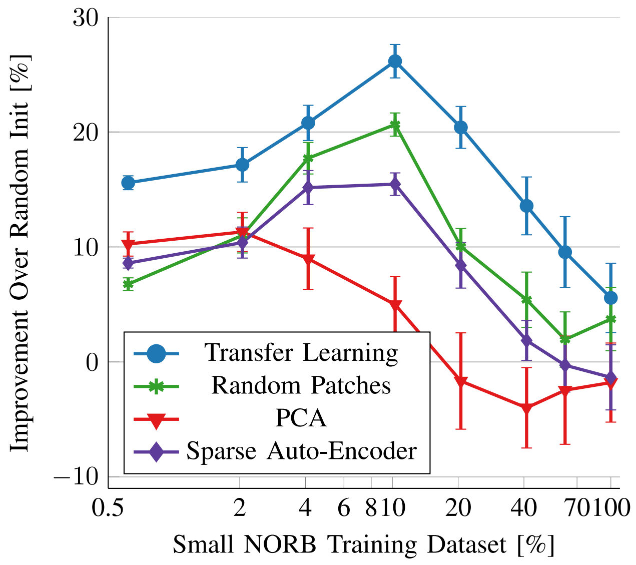
A. Visualization

Before presenting results in terms of classification performance, we will first show the feature representations that were learned from the respective initialization techniques. For each combination of dataset and learning strategy, five randomly chosen filter kernels are depicted in Fig. 3. To investigate how the features evolve when more samples are added to the training dataset, we refined the same starting point with different dataset sizes. The starting point of the supervised refinement is depicted in the bottom row, the respective features after network convergence are shown in the middle row (100 samples per class) and in the top row (all samples).

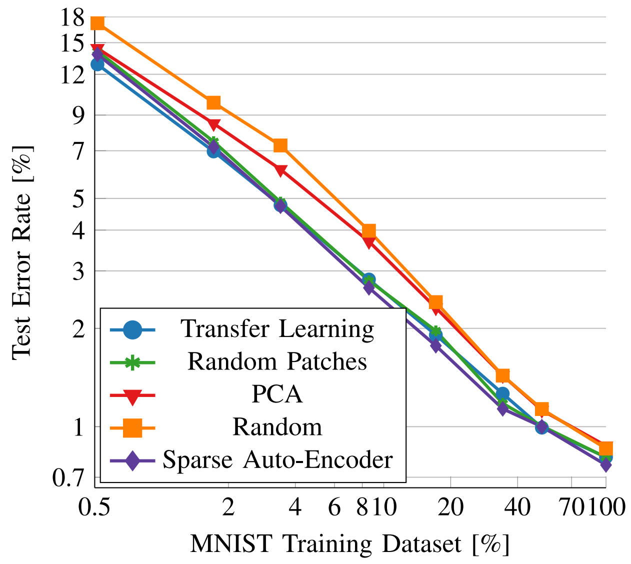
For the randomly initialized networks we can perceive the common behavior: Starting from pure random filters, the features converge to blob/edge-like detectors. The more samples were added to the training dataset, the less noisy the detectors are. The sparse auto-encoder produces Gabor-like filter kernels, but after refinement there is hardly a visual



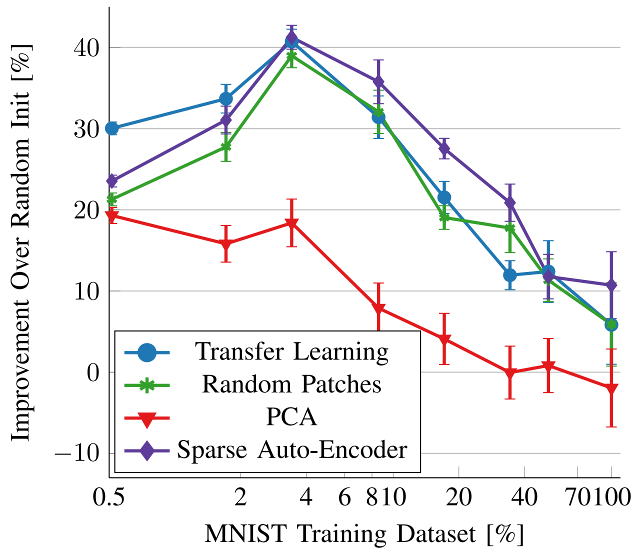
(a) Absolute test error evaluation on the Small NORB dataset.



(b) Improvement over the random initialization performance.



(c) Absolute test error evaluation on the MNIST dataset.



(d) Improvement over the random initialization performance.

Fig. 4: Classification results on the Small NORB (a), (b) and the MNIST (c), (d) dataset for different dataset sizes and initialization schemes. In order to retain visibility only the mean test error is reported in (a), (c). Error bars in (b), (d) indicate one standard deviation distance to the mean.

difference to the original filter. The same behavior can be observed for the features obtained using the transfer-learning initialization scheme They only evolve slightly during refinement. In contrast to the smooth edges of the other learned starting points, the patch initialization has rather strong edges. These also do not change much during refinement. The PCA initialization also only evolves slightly during refinement, even the high-frequency components prevail after supervised training. It can be seen that these initializations are indeed converging to features that are adapted to the task they were refined on when the features are embedded into two-dimensional space. We used t-Distributed Stochastic Embedding (t-SNE) [39] with the default parameter set to visualize the features in 2D (simple PCA has lead to similar results). Figure 5 shows the five MNIST transfer learning features from Fig. 3 when refined with different dataset sizes. Every stroke belongs to one feature, the dataset size is color-coded. The trajectories towards the largest training dataset size indicate that given a feature starting point, supervised refinement leads to convergence in a local neighborhood.

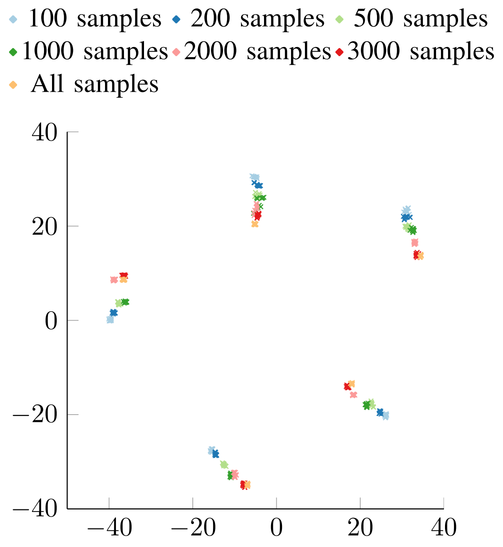


Fig. 5: t-SNE 2D visualization of five transfer learning MNIST features after refinement. Best viewed in color.

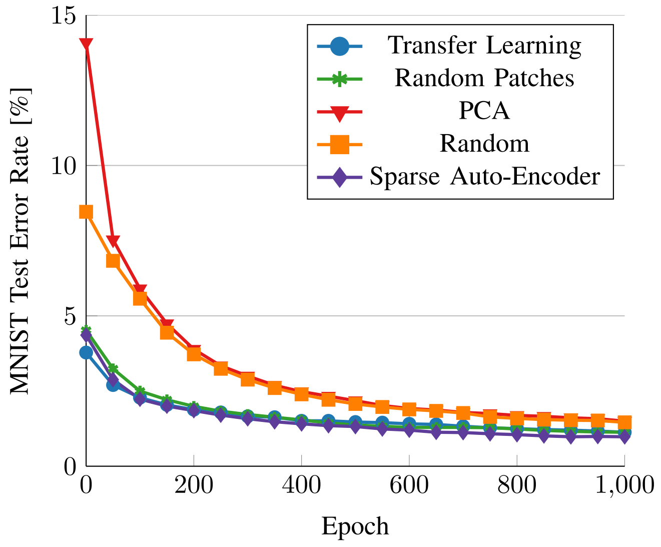


Fig. 6: Test error rate [%] as a function of the training epoch during refinement on the full MNIST training dataset.

B. Classification Performance

The classification results for all varied parameters are presented in this section. We report the test errors on both datasets as well as the relative improvement over the random initializations (Fig. 4). The major finding of our work is that initializing CNNs prior to supervised refinement does systematically lead to improved results when dealing with small datasets. For example when training on 3% of the MNIST dataset, an initialization with auto-encoder weights lowers the test error rate by 41% on average. The performance of the individual initialization schemes has shown itself to be dataset-dependent. On the MNIST dataset, no clear order of training strategies is established. However, when training on the full dataset, the auto-encoder pre-training gains over 10% compared to the random initialization.

On the other hand, on the Small NORB dataset the transfer learning approach clearly outperforms both the auto-encoder and the random patch initialization. This shows that knowledge transfer can help to compensate a lack of training data even on unrelated datasets. On the other hand, transferring knowledge does not necessary has the same impact in both directions. While the transfer from Small NORB to MNIST did not prove better than the competing methods, the reverse transfer outperforms the other approaches consistently. Surprisingly, the very simple random patch initialization performed well despite its simplicity. Admittedly, it has not shown to have the best performance on both datasets but it nevertheless ranged in between the competing learning strategies. It consistently outperformed the random initialization, even on the full dataset. The performance of the PCA scheme is mostly inferior to the competing initialization methods, but it manages to outperform the random initialization at least on the very small datasets (less then 10% of the original dataset). On larger Small NORB datasets, the PCA initialization harms the classification performance.

Faster convergence during training has been reported only for transfer learning so far [7]. Our results show that this holds as well for the other initialization schemes, with the exception of PCA. Especially in the beginning of the training, the CNNs initialized randomly and using PCA converge and reduce the test error much slower compared to the other learning strategies (see Fig. 6).

VII. Discussion

The results of our work are consistent with the literature, in particular that both unsupervised pre-training and transfer learning help to learn CNNs when training data is scarce. On the other hand, when sufficient labeled data is supplied the difference between the pre-trained and the randomly started networks diminish. In the case of transfer learning, we have shown that knowledge transfer between CNNs does not need strongly semantically related datasets. Nevertheless, the direction of the knowledge transfer does play a role. Astonishingly, the very simple random patch initialization scheme performed well on both datasets and on all dataset sizes. When one looks for a CNN initialization scheme without wishing to commit to the effort of transfer learning or auto-encoder training, the simple patch initialization is a promising alternative to randomly drawn weights.

VIII. Conclusion

We have shown in this paper that a well chosen initialization starting point for a supervised refinement is substantially superior to pure random initialization when training data is scarce. For this, we have compared the performance of unsupervised feature learning and transfer learning against simple patch initialization and random weight initialization within the same setup. Our results show that pre-training systematically improves generalization capabilities when handling datasets with few samples. Unfortunately, the choice of a pre-training method is highly dependent on the dataset. On the Small NORB dataset, the knowledge transfer from MNIST was consistently the best performing method, while on MNIST no clear order of training strategies could be achieved. Surprisingly, the unsupervised baseline method of randomly sampled patches from the training dataset consistently improves the generalization performance with respect to random initialization on both datasets. Further, it even beats more sophisticated methods in some scenarios. In general, we can conclude that proper weight initialization helps, but the particular method is not that important, as even simple heuristics can push the performance significantly.

References

[1] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[2] K. Jarrett, K. Kavukcuoglu, M. A. Ranzato, and Y. LeCun, "What is the best multi-stage architecture for object recognition?" in *Proc. of ICCV*, 2009.

[3] P. Y. Simard, D. Steinkraus, and J. C. Platt, "Best practices for convolutional neural networks applied to visual document analysis," in *Proc. of ICDAR*, 2003.

[4] G. Nguyen, S. Phung, and A. Bouzerdoum, "Reduced training of convolutional neural networks for pedestrian detection," in *Proc. of ICITA*, 2009.

[5] S. Thrun, "Is learning the n-th thing any easier than learning the first?" in *Proc. of NIPS*, 1996.

[6] S. Gutstein, O. Fuentes, and E. Freudenthal, "Knowledge transfer in deep convolutional neural nets," *International Journal on Artificial Intelligence Tools*, vol. 17, no. 3, p. 555, 2008.

[7] D. Cireşan, U. Meier, and J. Schmidhuber, "Transfer learning for latin and chinese characters with deep neural networks," in *Proc. of IJCNN*, 2012.

[8] A. Ahmed, W. KaiYu, Y. Gong, and E. Xing, "Training hierarchical feed-forward visual recognition models using transfer learning from pseudo-tasks," in *Proc. of ECCV*, 2008.

[9] D. Erhan, Y. Bengio, A. Courville, P. Manzagol, P. Vincent, and S. Bengio, "Why does unsupervised pre-training help deep learning?" *Journal of Machine Learning Research*, vol. 11, pp. 625–660, 2010.

[10] M. Ranzato, C. Poultney, S. Chopra, and Y. LeCun, "Efficient Learning of Sparse Representations with an Energy-based Model," in *Proc. of NIPS*, 2006.

[11] P. Vincent, H. Larochelle, I. Lajoie, Y. Bengio, and P.-A. Manzagol, "Stacked Denoising Autoencoders: Learning Useful Representations in a Deep Network with a Local Denoising Criterion," *Journal of Machine Learning Research*, vol. 11, pp. 3371–3408, 2010.

[12] G. Hinton, S. Osindero, and Y. Teh, "A fast learning algorithm for deep belief nets," *Neural Computation*, vol. 18, no. 7, pp. 1527–1554, 2006.

[13] Y. Bengio, P. Lamblin, D. Popovici, and H. Larochelle, "Greedy Layer-Wise Training of Deep Networks," in *Proc. of NIPS*, 2007.

[14] M. Ranzato, F. Huang, Y. Boureau, and Y. LeCun, "Unsupervised learning of invariant feature hierarchies with applications to object recognition," in *Proc. of CVPR*, 2007.

[15] J. Masci, U. Meier, D. Cireşan, and J. Schmidhuber, "Stacked convolutional auto-encoders for hierarchical feature extraction," in *Proc. of ICANN*, 2011.

[16] D. Hubel and T. Wiesel, "Receptive fields, binocular interaction and functional architecture in the cat's visual cortex," *The Journal of Physiology*, vol. 160, no. 1, p. 106, 1962.

[17] A. Krizhevsky, I. Sutskever, and G. Hinton, "ImageNet classification with deep convolutional neural networks," in *Proc. of NIPS*, 2012.

[18] Y. LeCun, F. Huang, and L. Bottou, "Learning methods for generic object recognition with invariance to pose and lighting," in *Proc. of CVPR*, 2004.

[19] D. Cireşan, U. Meier, J. Masci, and J. Schmidhuber, "A committee of neural networks for traffic sign classification," in *Proc. of IJCNN*, 2011.

[20] Y. LeCun, L. Bottou, G. Orr, and K. Müller, "Efficient backprop," *Neural Networks: Tricks of the Trade*, pp. 9–50, 1998.

[21] D. Scherer, A. Müller, and S. Behnke, "Evaluation of pooling operations in convolutional architectures for object recognition," in *Proc. of ICANN*, 2010.

[22] Y. LeCun and C. Cortes, "The MNIST Database of Handwritten Digits," 1998, `http://yann.lecun.com/exdb/mnist`.

[23] Q. Le, J. Ngiam, Z. Chen, D. Chia, P. Koh, and A. Ng, "Tiled convolutional neural networks," in *Proc. of NIPS*, 2010.

[24] C. Bishop, *Neural networks for pattern recognition*. Oxford Clarendon Press, 1995.

[25] T. Denoeux and R. Lengellé, "Initializing back propagation networks with prototypes," *Neural Networks*, vol. 6, no. 3, pp. 351–363, 1993.

[26] A. Coates and A. Y. Ng, "The importance of encoding versus training with sparse coding and vector quantization," in *Proc. of ICML*, 2011.

[27] I. T. Jolliffe, *Principal Component Analysis*. Springer, 1986.

[28] A. Hyvärinen, J. Hurri, and P. Hoyer, *Natural Image Statistics A probabilistic approach to early computational vision*. Springer, 2009.

[29] R. Raina, A. Battle, H. Lee, B. Packer, and A. Ng, "Self-taught learning: Transfer learning from unlabeled data," in *Proc. of ICML*, 2007.

[30] E. Oja, "A Simplified Neuron Model as a Principal Component Analyzer," *Journal of Mathematical Biology*, vol. 15, no. 3, pp. 267–273, 1982.

[31] T. D. Sanger, "Optimal Unsupervised Learning in a Single-Layer Linear Feedforward Neural Network," *Neural Networks*, vol. 2, no. 6, pp. 459–473, 1989.

[32] G. Hinton and R. Salakhutdinov, "Reducing the Dimensionality of Data with Neural Networks," *Science*, vol. 313, no. 5786, pp. 1527–1554, 2006.

[33] M. Thom, R. Schweiger, and G. Palm, "Supervised matrix factorization with sparseness constraints and fast inference," in *Proc. of IJCNN*, 2011.

[34] D. Hubel and T. Wiesel, "Receptive fields of single neurones in the cat's striate cortex," *The Journal of Physiology*, vol. 148, no. 3, pp. 574–591, 1959.

[35] B. Olshausen and D. Field, "Sparse coding of sensory inputs," *Current Opinion in Neurobiology*, vol. 14, no. 4, pp. 481–487, 2004.

[36] P. Lennie, "The Cost of Cortical Computation," *Current Biology*, vol. 13, no. 6, pp. 493–497, 2003.

[37] W. B. Levy and R. A. Baxter, "Energy Efficient Neural Codes," *Neural Computation*, vol. 8, no. 3, pp. 531–543, 1996.

[38] P. O. Hoyer, "Non-negative matrix factorization with sparseness constraints," *Journal of Machine Learning Research*, vol. 5, pp. 1457–1469, 2004.

[39] L. van der Maaten and G. Hinton, "Visualizing data using t-SNE," *Journal of Machine Learning Research*, vol. 9, pp. 2579–2605, 2008.