Learning by Association
A versatile semi-supervised training method for neural networks

Philip Haeusser[1,2] Alexander Mordvintsev[2] Daniel Cremers[1]

[1]Dept. of Informatics, TU Munich [2]Google, Inc.

{haeusser, cremers}@in.tum.de moralex@google.com

Abstract

In many real-world scenarios, labeled data for a specific machine learning task is costly to obtain. Semi-supervised training methods make use of abundantly available unlabeled data and a smaller number of labeled examples. We propose a new framework for semi-supervised training of deep neural networks inspired by learning in humans. "Associations" are made from embeddings of labeled samples to those of unlabeled ones and back. The optimization schedule encourages correct association cycles that end up at the same class from which the association was started and penalizes wrong associations ending at a different class. The implementation is easy to use and can be added to any existing end-to-end training setup. We demonstrate the capabilities of learning by association on several data sets and show that it can improve performance on classification tasks tremendously by making use of additionally available unlabeled data. In particular, for cases with few labeled data, our training scheme outperforms the current state of the art on SVHN. We also show how to apply this training scheme to the task of domain adaptation, surpassing current state-of-the-art results.

1. Introduction

A child is able to learn new concepts quickly and without the need for millions examples that are pointed out individually. Once a child has seen one dog, she or he will be able to recognize other dogs and becomes better at recognition with subsequent exposure to more variety.

In terms of training computers to perform similar tasks, deep neural networks have demonstrated superior performance among machine learning models ([20, 18, 10]). However, these networks have been trained dramatically differently from a learning child, requiring labels for every training example, following a purely supervised training scheme. Neural networks are defined by huge amounts of parameters to be optimized. Therefore, a plethora of labeled training data is required, which might be costly and time consuming to obtain. It is desirable to train machine learning models without labels (unsupervisedly) or with only some fraction of the data labeled (semi-supervisedly).

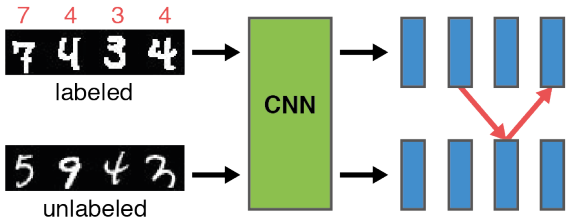

Figure 1. Learning by association. A network (green) is trained to produce embeddings (blue) that have high similarities if belonging to the same class. A differentiable association cycle (red) from embeddings of labeled (A) to unlabeled (B) data and back is used to evaluate the association.

Recently, efforts have been made to train neural networks in an unsupervised or semi-supervised manner yielding promising results. However, most of these methods require a trick to generate training data, such as sampling patches from an image for context prediction [6] or generating surrogate classes [7, 22, 13]. In other cases, semi-supervised training schemes require non trivial additional architectures such as generative adversarial networks [9] or a decoder part [39].

We propose a novel training method that follows an intuitive approach: learning by association (Figure 1). We feed a batch of labeled and a batch of unlabeled data through a network, producing embeddings for both batches. Then, an imaginary walker is sent from samples in the labeled batch to samples in the unlabeled batch. The transition follows a probability distribution obtained from the similarity of the respective embeddings which we refer to as an *association*. In order to evaluate whether the association makes sense, a

second step is taken back to the labeled batch - again guided by the similarity between the embeddings. It is now easy to check if the cycle ended at the same class from which it was started. We want to maximize the probability of consistent cycles, i.e., walks that return to the same class. Hence, the network is trained to produce embeddings that capture the essence of the different classes, leveraging unlabeled data. In addition, a classification loss can be specified, encouraging embeddings to generalize to the actual target task.

The association operations are fully differentiable, facilitating end-to-end training of arbitrary network architectures. Any existing classification network can be extended by our customized loss function.

In summary, our key contributions are:

- A novel yet simple training method that allows for semi-supervised end-to-end training of arbitrary network architectures. We name the method "associative learning".
- An open-source TensorFlow implementation[1] of our method that can be used to train arbitrary network architectures.
- Extensive experiments demonstrating that the proposed method improves performance by up to 64% compared to the purely supervised case.
- Competitive results on MNIST and SVHN, surpassing state of the art for the latter when only a few labeled samples are available.
- State-of-the-art results for domain adaptation (SVHN $\rightarrow$ MNIST and Synthetic Digits $\rightarrow$ SVHN).

2. Related Work

The challenge of harnessing unlabeled data for training of neural networks has been tackled using a variety of different methods. Although this work follows a semi-supervised approach, it is in its motivation also related to purely unsupervised methods. A third category of related work is constituted by generative approaches.

2.1. Semi-supervised training

The semi-supervised training paradigm has not been among the most popular methods for neural networks in the past. It has been successfully applied to SVMs [14] where unlabeled samples serve as additional regularizers in that decision boundaries are required to have a broad margin also to unlabeled samples.

One training scheme applicable to neural nets is to bootstrap the model with additional labeled data obtained from the model's own predictions. [22] introduce pseudo-labels for unlabeled samples which are simply the class with the maximum predicted probability. Labeled and unlabeled samples are then trained on simultaneously. In combination with a denoising auto-encoder and dropout, this approach yields competitive results on MNIST.

Other methods add an auto-encoder part to an existing network with the goal of enforcing efficient representations ([27] [37] [39]).

Recently, [30] introduced a regularization term that uses unlabeled data to push decision boundaries of neural networks to less dense areas of decision space and enforces mutual exclusivity of classes in a classification task. When combined with a cost function that enforces invariance to random transformations as in [31], state-of-the-art results on various classification tasks can be obtained.

2.2. Purely unsupervised training

Unsupervised training is obviously more general than semi-supervised approaches. It is, however, important to differentiate the exact purpose. While semi-supervised training allows for a certain degree of guidance as to what the network learns, the usefulness of unsupervised methods highly depends on the design of an appropriate cost function and balanced data sets. For exploratory purposes, it might be desirable that representations become more fine grained for different suptypes of one class in the data set. Conversely, if the ultimate goal is classification, invariance to this very phenomenon might be more preferable.

[12] propose to use Restricted Boltzmann Machines ([33]) to pre-train a network layer-wise with unlabeled data in an auto-encoder fashion.

[11][19][39] build a neural network upon an auto-encoder that acts as a regularizer and encourages representations that capture the essence of the input.

A whole new category of unsupervised training is to generate surrogate labels from data. [13] employ clustering methods that produce weak labels.

[7] generate surrogate classes from transformed samples from the data set. These transformations have hand-tuned parameters making it non-trivial to ensure they are capable of representing the variations in an arbitrary data set.

In the work of [6], context prediction is used as a surrogate task. The objective for the network is to predict the relative position of two randomly sampled patches of an image. The size of the patches needs to be manually tuned such that parts of objects in the image are not over- or undersampled.

[34] employ a multi-layer LSTM for unsupervised image sequence prediction/reconstruction, leveraging the temporal dimension of videos as the context for individual frames.

2.3. Generative Adversarial Nets (GANs)

The introduction of generative adversarial nets (GANs) [9] enabled a new discipline in unsupervised training. A

[1] https://git.io/vyzrl

generator network (G) and a discriminator network (D) are trained jointly where the G tries to generate images that look as if drawn from an unlabeled data set, whereas D is supposed to identify the difference between real samples and generated ones. Apart from providing compelling visual results, these networks have been shown to learn useful hierarchical representations [26].

[32] presents improvements in designing and training GANs, in particular, these authors achieve state-of-the-art results in semi-supervised classification on MNIST, CIFAR-10 and SVHN.

3. Learning by association

A general assumption behind our work is that good embeddings will have a high similarity if they belong to the same class. We want to optimize the parameters of a CNN in order to produce good embeddings, making use of both labeled and unlabeled data. A batch of labeled and unlabeled images (A_{img} and B_{img}, respectively) is fed through the CNN, resulting in embedding vectors (A and B). We then imagine a walker going from A to B according to the mutual similarities, and back. If the walker ended up at the same class as he started from, the walk is correct. The general scheme is depicted in Figure 1.

3.1. Mathematical formulation

The goal is to maximize the probability for correct walks from A to B and back to A, ending up at the same class. A and B are matrices whose rows index the samples in the batches. Let's define the **similarity between embeddings A and B** as

$$M_{ij} := A_i \cdot B_j \tag{1}$$

Note that the dot product could in general be replaced by any other similarity metric such as Euclidean distance. In our experiments, the dot product worked best in terms of convergence. Now, we transform these similarities into **transition probabilities from A to B** by softmaxing M over columns:

$$\begin{aligned} P_{ij}^{ab} = P(B_j|A_i) :=& (\text{softmax}_{\text{cols}}(M))_{ij} \\ =& \exp(M_{ij}) / \sum_{j'} \exp(M_{ij'}) \end{aligned} \tag{2}$$

Conversely, we get the transition probabilities in the other direction, P_{ba}, by replacing M with M^T. We can now define the **round trip probability** of starting at A_i and ending up at A_j:

$$\begin{aligned} P_{ij}^{aba} :=& (P^{ab}P^{ba})_{ij} \\ =& \sum_k P_{ik}^{ab} P_{kj}^{ba} \end{aligned} \tag{3}$$

Finally, the **probability for correct walks** becomes

$$P(\text{correct walk}) = \frac{1}{|A|} \sum_{i \sim j} P_{ij}^{aba} \tag{4}$$

where $i \sim j \Leftrightarrow \text{class}(A_i) = \text{class}(A_j)$.

We define multiple losses that encourage intuitive goals. These losses can be combined, as discussed in Section 4.

$$\mathcal{L}_{\text{total}} = \mathcal{L}_{\text{walker}} + \mathcal{L}_{\text{visit}} + \mathcal{L}_{\text{classification}} \tag{5}$$

Walker loss. The goal of our association cycles is consistency. A walk is consistent when it ends at a sample with the same class as the starting sample. This loss penalizes incorrect walks and encourages a uniform probability distribution of walks to the correct class. The uniform distribution models the idea that it is permitted to end the walk at a different sample than the starting one, as long as both belong to the same class. The walker loss is defined as the cross-entropy H between the uniform target distribution of correct round-trips T and the round-trip probabilities P^{aba}.

$$\mathcal{L}_{\text{walker}} := H(T, P^{aba}) \tag{6}$$

with the uniform target distribution

$$T_{ij} := \begin{cases} 1/\#\text{class}(A_i) & \text{class}(A_i) = \text{class}(A_j) \\ 0 & \text{else} \end{cases} \tag{7}$$

where $\#\text{class}(A_i)$ is the number of occurrences of $\text{class}(A_i)$ in A.

Visit loss. There might be samples in the unlabeled batch that are difficult, such as a badly drawn digit in MNIST. In order to make best use of all unlabeled samples, it should be beneficial to "visit" all of them, rather than just making associations among "easy" samples. This encourages embeddings that generalize better. The visit loss is defined as the cross-entropy H between the uniform target distribution V and the visit probabilities P^{visit}. If the unsupervised batch contains many classes that are not present in the supervised one, this regularization can be detrimental and needs to be weighted accordingly.

$$\mathcal{L}_{\text{visit}} := H\left(V, P^{\text{visit}}\right) \tag{8}$$

where the visit probability for examples in B and the uniform target distribution are defined as follows:

$$P_j^{\text{visit}} := \langle P_{ij}^{ab} \rangle_i \tag{9}$$

$$V_j := 1/|B| \tag{10}$$

Classification loss. So far, only the creation of embeddings has been addressed. These embeddings can easily be mapped to classes by adding an additional fully connected

layer with softmax and a cross-entropy loss on top of the network. We call this loss classification loss. This mapping to classes is necessary to evaluate a network's performance on a test set. However, convergence can also be reached without it.

3.2. Implementation

The total loss $\mathcal{L}_{\text{total}}$ is minimized using Adam [16] with the suggested default settings. We applied random data augmentation where mentioned in Section 4. The training procedure is implemented end-to-end in TensorFlow [1] and the code is publicly available.

4. Experiments

In order to demonstrate the capabilities of our proposed training paradigm, we performed different experiments on various data sets. Unless stated otherwise, we used the following network architecture with batch size 100 for both labeled batch A (10 samples per class) and unlabeled batch B:

$$\begin{aligned} & C(32,3) \rightarrow C(32,3) \rightarrow P(2) \\ \rightarrow\ & C(64,3) \rightarrow C(64,3) \rightarrow P(2) \\ \rightarrow\ & C(128,3) \rightarrow C(128,3) \rightarrow P(2) \rightarrow FC(128) \end{aligned}$$

Here, $C(n,k)$ stands for a convolutional layer with n kernels of size $k \times k$ and stride 1. $P(k)$ denotes a max-pooling layer with window size $k \times k$ and stride 1. $FC(n)$ is a fully connected layer with n output units.

Convolutional and fully connected layers have exponential linear units (elu) activation functions [3] and an additional L2 weight regularizer with weight 10^{-4} applied.

There is an additional FC layer, mapping the embedding to the logits for classification after the last FC layer that produces the embedding, i.e., $FC(10)$ for 10 classes.

4.1. MNIST

The MNIST data set [21] is a benchmark containing handwritten digits for supervised classification. Mutual exclusivity regularization with transformations ([28]) have previously set the state of the art among semi-supervised deep learning methods on this benchmark. We trained the simple architecture mentioned above with our approach with all three losses from Section 3.1 and achieved competitive results as shown in Table 1. We have not even started to explore sophisticated additional regularization schemes that might further improve our results. The main point of these first experiments was to test how quickly one can achieve competitive results with a vanilla architecture, purely by adding our proposed training scheme. In the following, we explore some interesting, easily reproducible properties.

4.1.1 Evolution of associations

The untrained network is already able to make some first associations based on the produced embeddings. However, many wrong associations are made and only a few samples in the unsupervised batch (B) are visited: those most similar to the examples in the supervised batch (A). As training progresses, these associations get better. The visit loss ensures that all samples in B are visited with equal probability. Figure 2 shows this evolution. The original samples for a setup with 2 labeled samples per class are shown where A is green and B is red. Associations are made top-down. Note that the second set of green digits is equal to the first ("round-trip"). The top graphic in Figure 2 shows visit probabilities at the beginning of training. Darker lines denote a higher probability (softmaxed dotproduct). The bottom graphic in Figure 2 shows associations after training has converged. This took 10k iterations during which only the same 20 labeled samples were used for A and samples for B were drawn randomly from the rest of the data set, ignoring labels.

4.1.2 Confusion analysis

Even after training has converged, the network still makes mistakes. These mistakes can, however, be explained. Figure 3 shows a confusion matrix for the classification task. On the left side, all samples from the labeled set (A) are shown (10 per class). Those samples that are classified incorrectly express features that are not present in the supervised training set, e.g. "7" with a bar in the middle (mistaken for "2") or "4" with a closed loop (mistaken for "9"). Obviously, A needs to be somewhat representative for the data set, as is usually the case for machine learning tasks.

4.2. STL-10

STL-10 is a data set of RGB images from 10 classes [4]. There are 5k labeled training samples and 100k unlabeled training images from the same 10 classes and additional classes not present in the labeled set. For this task we modified the network architecture slightly as follows:

$$\begin{aligned} & C(32,3) \rightarrow C(64,3,\text{stride=2}) \rightarrow P(3) \\ \rightarrow\ & C(64,3) \rightarrow C(128,3) \rightarrow P(2) \\ \rightarrow\ & C(128,3) \rightarrow C(256,3) \rightarrow P(2) \rightarrow FC(128) \end{aligned}$$

As a preprocessing step, we apply various forms of data augmentation to all samples fed though the net. In particular, random cropping, changes in brightness, saturation, hue and small rotations.

We ran training using 100 randomly chosen samples per class from the labeled training set for A (i.e. we used only 20% of the labeled training images) and achieved an accuracy on the test set of 81%. As this is not exactly following

Method	# labeled samples		
	100	1000	All
Ladder, full [28]	1.06 (0.37)	0.84 (0.08)	0.57 (0.02)
Improved GAN [32]	0.93 ($\pm$ 0.07)	-	-
Mutual Exclusivity + Transform. [31]	**0.55 (0.16)**	-	**0.27 (0.02)**
Ours	0.89 (0.08)	0.74 (0.03)	0.36 (0.03)

Table 1. Results on MNIST. Error (%) on the test set (lower is better). Standard deviations in parentheses.

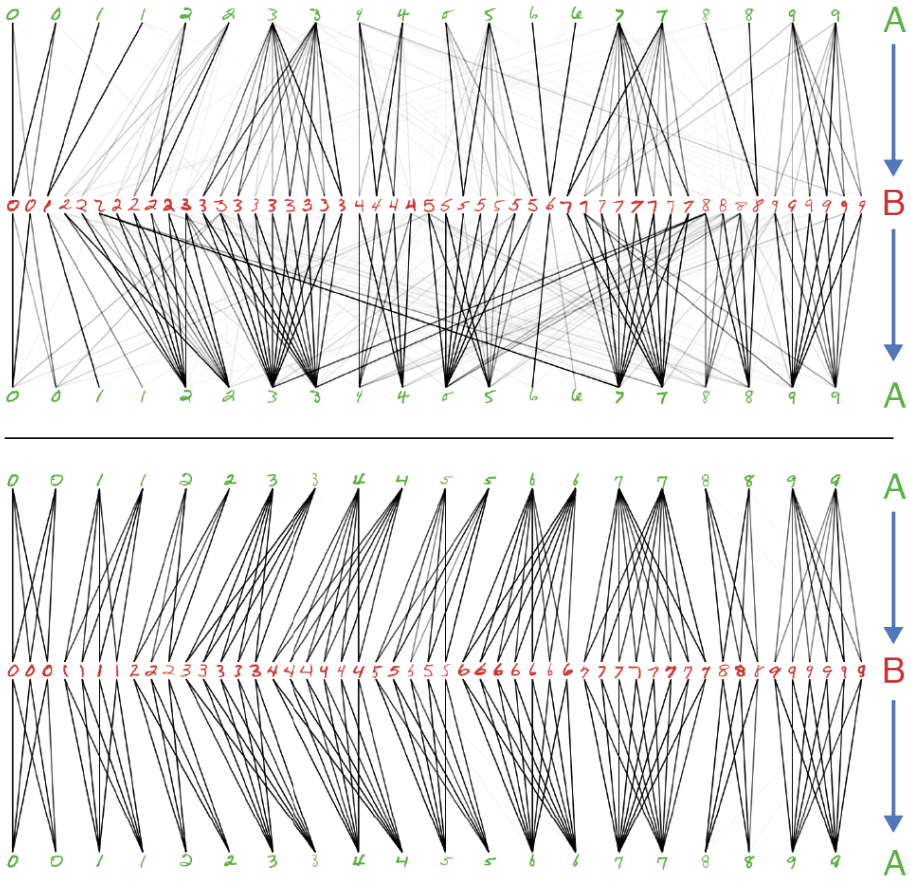

Figure 2. Evolution of associations. Top: in the beginning of training, after a few iterations. Bottom: after convergence. Green digits are the supervised set (A) and red digits are samples from the unsupervised set (B).

the testing protocol suggested by the data set creators, we do not want to claim state of the art for this experiment but do consider it a promising result. [13] achieved 76.3% following the proposed protocol.

The unlabeled training set contains many other classes and it is interesting to examine the trained net's associations with them. Figure 4 shows the 5 nearest neighbors (cosine distance) for samples from the unlabeled training set. The cosine similarity is shown in the top left corner of each association. Note that these numbers are not softmaxed. Known classes (top two rows) are mostly associated correctly, whereas new classes (bottom two rows) are associated with other classes, yet exposing interesting connections: The fin of a dolphin reminds the net of triangularly shaped objects such as the winglet of an airplane wing. A meerkat looking to the right is associated with a dog looking in the same direction or with a racoon with dark spots around the eyes. Unfortunately, embeddings of classes not present in the labeled training set do not seem to group together well; rather, they tend to be close to known class representations.

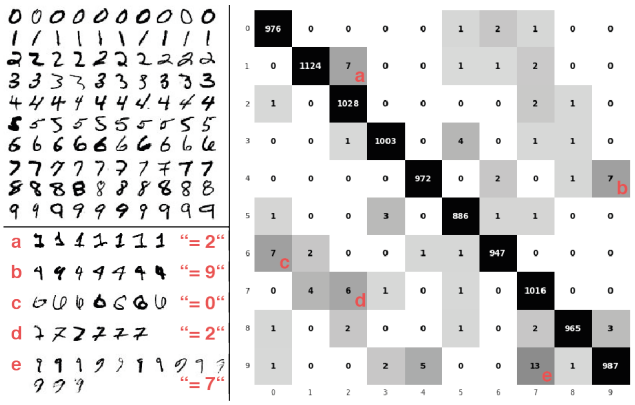

Figure 3. MNIST classification. Top left: All labeled samples that were used for training. Right: Confusion matrix with mistakes that were made. Test error: 0.96%. Bottom left: Misclassified examples from the test.

Figure 4. Nearest neighbors for samples from the unlabeled training set. The far left column shows the samples, the 5 other columns are the nearest neighbors in terms of cosine distance (which is shown in the top left corners of the pictures).

4.3. SVHN

The Street View House Numbers (SVHN) data set [25] contains digits extracted from house numbers in Google Street View images. We use the format 2 variant where digits are cropped to 32x32 pixels. This variant is similar to MNIST in structure, yet the statistics are a lot more complex and richer in variation. The train and test subsets contain 73,257 and 26,032 digits, respectively.

We performed the same experiments as for MNIST with the following architecture:

$$C(32,3) \to C(32,3) \to C(32,3) \to P(2)$$
$$\to C(64,3) \to C(64,3) \to C(64,3) \to P(2)$$
$$\to C(128,3) \to C(128,3) \to C(128,3) \to P(2) \to FC(128)$$

Data augmentation is achieved by applying random affine transformations and Gaussian blurring to model the variations evident in SVHN.

4.4. Effect of adding unlabeled data

In order to quantify how useful it is to add unlabeled data to the training process with our approach, we trained the same network architecture with different amounts of labeled and unlabeled data. For the case of no unlabeled data, only $\mathcal{L}_{\text{classification}}$ is active. In the other cases where labeled data is present, we optimize $\mathcal{L}_{\text{total}}$. We ran the nets on 10 randomly chosen subsets of the data and report median and standard deviation.

Table 3 shows results on SVHN. We used the (labeled) SVHN training set as data corpus from which we drew randomly chosen subsets as labeled and unlabeled sets. There might be overlaps between both of these sets, which would mean that the reported error rates can be seen as upper bounds.

Let's consider the case of fully supervised training. This corresponds to the far left column in Table 3. Not surprisingly, the more labeled samples are used, the lower the error on the test set gets.

We now add unlabeled data. For a setup with only 20 labeled samples (2 per class), the baseline is an error rate of 83.97% for 0 additional unlabeled samples. Performance deteriorates as more unlabeled samples are added. This setting seems to be pathological: depending on the data set, there is a minimum number of samples required for successful generalization.

In all other scenarios with a greater number of labeled samples, the general pattern we observed is that performance improves with greater amounts of unlabeled data. This indicates that it is indeed possible to boost a network's performance just by adding unlabeled data using the proposed associative learning scheme. For example, in the case of 500 labeled samples, it was possible to decrease the test error by 64.8% (from 17.75% to 6.25%).

A particular case occurs when all data is used in the labeled batch (last row in Table 3): Here, all samples in the unlabeled set are also in the labeled set. This means that the unlabeled set does not contain new information. Nevertheless, employing associative learning with unlabeled data improves the network's performance. $\mathcal{L}_{\text{walker}}$ and $\mathcal{L}_{\text{visit}}$ act as a beneficial regularizer that enforces similarity of embeddings belonging to the same class. This means that associative learning can also help in situations where a purely supervised training scheme has been used, without the need for additional unlabeled data.

4.5. Effect of visit loss

Section 3.1 introduces different losses. We wanted to investigate the effects of our proposed visit loss. To this end, we trained networks on different data sets and varied the

Method	# labeled samples		
	500	1000	2000
DGN [17]		36.02 (0.10)	
Virtual Adversarial [24]		24.63	
Auxiliary Deep Generative Model [23]		22.86	
Skip Deep Generative Model [23]		16.61 (0.24)	
Imporoved GAN [32]	18.44 (4.8)	8.11 (1.3)	6.16 (0.58)
Imporoved GAN (Ensemble) [32]		5.88 (1.0)	
Mutual Exclusivity + Transform.* [31]	9.62 (1.37)	**4.52 (0.40)**	**3.66 (0.14)**
Ours	**6.25 (0.32)**	5.14 (0.17)	4.60 (0.21)

Table 2. Results of comparable methods on SVHN. Error (%) on the test set (lower is better). Standard deviations in parentheses. *) Results provided by authors.

# labeled samples	# unlabeled samples			
	0	1000	20000	all
20	81.00 (3.01)	81.98 (2.58)	82.15 (1.35)	82.10 (1.91)
100	55.64 (6.54)	39.85 (7.19)	24.31 (7.19)	23.18 (7.41)
500	17.75 (0.65)	12.78 (0.99)	6.61 (0.32)	6.25 (0.32)
1000	10.92 (0.24)	9.10 (0.37)	5.48 (0.34)	5.14 (0.17)
2000	8.25 (0.32)	7.27 (0.43)	4.83 (0.15)	4.60 (0.21)
all	3.09 (0.06)	2.79 (0.02)	2.80 (0.03)	2.69 (0.05)

Table 3. Results on SVHN with different amounts of (total) labeled/unlabeled training data. Error (%) on the test set (lower is better). Standard deviations in parentheses.

loss weights for $\mathcal{L}_{\text{visit}}$ keeping the loss weight for $\mathcal{L}_{\text{classification}}$ and $\mathcal{L}_{\text{walker}}$ constant. Table 4 shows the results. Worst performance was obtained with no visit loss. For MNIST, visit loss is crucial for successful training. For SVHN, a moderate loss weight of about 0.25 leads to best performance. If the visit loss weight is too high, the effect seems to be over regularization of the network.. This suggests that the visit loss weight needs to be adapted according to the variance within a data set. If the distributions of samples in the (finitely sized) labeled and unlabeled batches are less similar, the visit loss weight should be lower.

4.6. Domain adaptation

A test for the efficiency of representations is to apply a model to the task of domain adaptation (DA) [29]. The general idea is to train a model on data from a source domain and then adapt it to similar but different data from a target domain.

In the context of neural networks, DA has mostly been achieved by either *fine-tuning* a network on the target domain after training it on the source domain ([36, 15]), or by designing a network with multiple outputs for the respective domains ([5, 38]), sometimes referred to as *dual outputs*.

As a first attempt at DA with associative learning, we propose the following procedure that is a mix of both fine-tuning and dual outputs: We first train a network on the source domain as described in Section 4. Then, we only exchange the unsupervised data set to the target domain data and continue training. Note that here, no labels from the target class are used at all at train time.

As a baseline example, we chose a network trained on SVHN. We fed labeled samples from SVHN (source domain) and unlabeled samples from MNIST (target domain) in the network with the architecture originally used for training on the source domain and fine-tuned it with our association based approach. No data augmentation was applied.

Initially, the network achieved an error of 18.56% on the MNIST test set which we found surprisingly low, considering that the network had not previously seen an MNIST digit. Some SVHN examples have enough similarity to MNIST that the network recognized a considerable amount of handwritten digits.

We then trained the network with both data sources as described above with 0.5 as weight for the visit loss. After 9k iterations the network reached an accuracy of 0.51% on

Data set	Visit loss weight			
	0	0.25	0.5	1
MNIST	5.68 (0.53)	1.17 (0.15)	**0.82** (0.12)	0.85 (0.04)
SVHN	7.91 (0.40)	**6.31** (0.20)	6.32 (0.07)	6.43 (0.26)

Table 4. Effect of visit loss. Error (%) on the resp. test sets (lower is better) for different values of visit loss weight. Reported are the medians of the minimum error rates throughout training with standard deviation in parentheses. Experiments were run with 1,000 randomly chosen labeled samples as supervised data set.

Data	Method	Domains (source → target)		
		SVHN → MNIST	MNIST → SVHN	Synth → SVHN
Source only	DA [8]	45.10		13.26
	DS [2]	40.8		13.3
	Ours	18.56	77.33	15.97
Adapted	DA [8]	26.15 (42.6%)		8.91 (79.7%)
	DS [2]	17.3 (58.3%)		8.8 (78.9%)
	Ours	**0.51 (99.3%)**	33.20 (59.0%)	**6.90 (67.6%)**
Target only	DA [8]	0.58		7.80
	DS [2]	0.5		7.6
	Ours	0.38	2.56	2.56

Table 5. Domain adaptation. Errors (%) on the target test sets (lower is better). "Source only" and "target only" refers to training only on the respective data set without domain adaptation. "DA" and "DS" stand for Domain-Adversarial Training and Domain Separation Networks, resp. The numbers in parentheses indicate how much of the gap between lower and upper bounds was covered.

the MNIST test set, which is a higher accuracy than what we reached when training a network with 100 or 1000 labeled samples from MNIST (cf. Section 4.1).

A more difficult scenario is the reverse case: MNIST → SVHN. We carried out the same experiment as described above but with swapped data sets and network architectures. In order to account for the fact that some SVHN numbers are black on white background, we randomly ($p = 0.5$) inverted to the MNIST samples as augmentation.

We repeated this procedure with a more realistic setting: We trained a network on a synthetic data set of digits rendered with different fonts provided by [8]. This kind of labeled data is easy and cheap to produce abundantly. We then ran domain adaptation on the much more difficult SVHN data set (without any data augmentation) and reached a test error of 6.90% after 5k iterations.

For comparison, [2] has been holding state of the art for domain adaptation employing domain separation networks. Table 5 contrasts their results with ours. Our first tentative training method for DA outperforms traditional methods by a large margin. We therefore conclude that learning by association is a promising training scheme that encourages efficient embeddings.

5. Conclusion

We have proposed a novel semi-supervised training scheme that is fully differentiable and easy to add to existing end-to-end settings. The key idea is to encourage cycle-consistent association chains from embeddings of labeled data to those of unlabeled ones and back. The code is publicly available. Although we have not employed sophisticated network architectures such as ResNet [10] or Inception [35], we achieve competitive results with simple networks trained with the proposed approach. We have demonstrated how adding unlabeled data improves results dramatically, in particular when the number of labeled samples is small, surpassing state of the art for SVHN with 500 labeled samples. A very promising experiment was to use our approach for domain adaptation where we outperform state of the art on SVHN → MNIST and Synthetic Digits → SVHN. Investigating the scalability to thousands of classes or maybe even completely different problems such as segmentation will be the subject of future research.

References

[1] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, et al. Tensorflow: Large-scale machine learning on heterogeneous distributed systems. *arXiv preprint arXiv:1603.04467*, 2016. 4

[2] K. Bousmalis, G. Trigeorgis, N. Silberman, D. Krishnan, and D. Erhan. Domain separation networks. *arXiv preprint arXiv:1608.06019*, 2016. 8

[3] D.-A. Clevert, T. Unterthiner, and S. Hochreiter. Fast and accurate deep network learning by exponential linear units (elus). *arXiv preprint arXiv:1511.07289*, 2015. 4

[4] A. Coates, H. Lee, and A. Y. Ng. An analysis of single-layer networks in unsupervised feature learning. *Ann Arbor*, 1001(48109):2, 2010. 4

[5] R. Collobert, J. Weston, L. Bottou, M. Karlen, K. Kavukcuoglu, and P. Kuksa. Natural language processing (almost) from scratch. *Journal of Machine Learning Research*, 12(Aug):2493–2537, 2011. 7

[6] C. Doersch, A. Gupta, and A. A. Efros. Unsupervised visual representation learning by context prediction. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1422–1430, 2015. 1, 2

[7] A. Dosovitskiy, J. T. Springenberg, M. Riedmiller, and T. Brox. Discriminative unsupervised feature learning with convolutional neural networks. In *Advances in Neural Information Processing Systems*, pages 766–774, 2014. 1, 2

[8] Y. Ganin, E. Ustinova, H. Ajakan, P. Germain, H. Larochelle, F. Laviolette, M. Marchand, and V. Lempitsky. Domain-adversarial training of neural networks. *Journal of Machine Learning Research*, 17(59):1–35, 2016. 8

[9] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative adversarial nets. In *Advances in Neural Information Processing Systems*, pages 2672–2680, 2014. 1, 2

[10] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. *arXiv preprint arXiv:1512.03385*, 2015. 1, 8

[11] I. Higgins, L. Matthey, X. Glorot, A. Pal, B. Uria, C. Blundell, S. Mohamed, and A. Lerchner. Early visual concept learning with unsupervised deep learning. *arXiv preprint arXiv:1606.05579*, 2016. 2

[12] G. Hinton. A practical guide to training restricted boltzmann machines. *Momentum*, 9(1):926, 2010. 2

[13] C. Huang, C. Change Loy, and X. Tang. Unsupervised learning of discriminative attributes and visual representations. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5175–5184, 2016. 1, 2, 5

[14] T. Joachims. Transductive inference for text classification using support vector machines. In *ICML*, volume 99, pages 200–209, 1999. 2

[15] Y. Kim. Convolutional neural networks for sentence classification. *arXiv preprint arXiv:1408.5882*, 2014. 7

[16] D. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014. 4

[17] D. P. Kingma, S. Mohamed, D. Jimenez Rezende, and M. Welling. Semi-supervised learning with deep generative models. In Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 27*, pages 3581–3589. Curran Associates, Inc., 2014. 7

[18] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 25*, pages 1097–1105. Curran Associates, Inc., 2012. 1

[19] Q. V. Le. Building high-level features using large scale unsupervised learning. In *2013 IEEE international conference on acoustics, speech and signal processing*, pages 8595–8598. IEEE, 2013. 2

[20] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998. 1

[21] Y. LeCun, C. Cortes, and C. J. Burges. The mnist database of handwritten digits, 1998. 4

[22] D.-H. Lee. Pseudo-label: The simple and efficient semi-supervised learning method for deep neural networks. In *Workshop on Challenges in Representation Learning, ICML*, volume 3, page 2, 2013. 1, 2

[23] L. Maaløe, C. K. Sønderby, S. K. Sønderby, and O. Winther. Auxiliary deep generative models. *arXiv preprint arXiv:1602.05473*, 2016. 7

[24] T. Miyato, S.-i. Maeda, M. Koyama, K. Nakae, and S. Ishii. Distributional smoothing by virtual adversarial examples. *arXiv preprint arXiv:1507.00677*, 2015. 7

[25] Y. Netzer, T. Wang, A. Coates, A. Bissacco, B. Wu, and A. Y. Ng. Reading digits in natural images with unsupervised feature learning. In *NIPS workshop on deep learning and unsupervised feature learning*, volume 2011, page 4. Granada, Spain, 2011. 6

[26] A. Radford, L. Metz, and S. Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv preprint arXiv:1511.06434*, 2015. 3

[27] M. Ranzato and M. Szummer. Semi-supervised learning of compact document representations with deep networks. In *Proceedings of the 25th international conference on Machine learning*, pages 792–799. ACM, 2008. 2

[28] A. Rasmus, M. Berglund, M. Honkala, H. Valpola, and T. Raiko. Semi-supervised learning with ladder networks. In *Advances in Neural Information Processing Systems*, pages 3546–3554, 2015. 4, 5

[29] K. Saenko, B. Kulis, M. Fritz, and T. Darrell. Adapting visual category models to new domains. In *European conference on computer vision*, pages 213–226. Springer, 2010. 7

[30] M. Sajjadi, M. Javanmardi, and T. Tasdizen. Mutual exclusivity loss for semi-supervised deep learning. In *2016 IEEE International Conference on Image Processing (ICIP)*, pages 1908–1912. IEEE, 2016. 2

[31] M. Sajjadi, M. Javanmardi, and T. Tasdizen. Regularization with stochastic transformations and perturbations for deep semi-supervised learning. *arXiv preprint arXiv:1606.04586*, 2016. 2, 5, 7

[32] T. Salimans, I. Goodfellow, W. Zaremba, V. Cheung, A. Radford, and X. Chen. Improved techniques for training gans. *arXiv preprint arXiv:1606.03498*, 2016. 3, 5, 7

[33] P. Smolensky. Information processing in dynamical systems: Foundations of harmony theory. Technical report, DTIC Document, 1986. 2

[34] N. Srivastava, E. Mansimov, and R. Salakhutdinov. Unsupervised learning of video representations using lstms. *CoRR, abs/1502.04681*, 2, 2015. 2

[35] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going deeper with convolutions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–9, 2015. 8

[36] S. Venugopalan, H. Xu, J. Donahue, M. Rohrbach, R. Mooney, and K. Saenko. Translating videos to natural language using deep recurrent neural networks. *arXiv preprint arXiv:1412.4729*, 2014. 7

[37] J. Weston, F. Ratle, H. Mobahi, and R. Collobert. Deep learning via semi-supervised embedding. In *Neural Networks: Tricks of the Trade*, pages 639–655. Springer, 2012. 2

[38] Z. Yang, R. Salakhutdinov, and W. Cohen. Multi-task cross-lingual sequence tagging from scratch. *arXiv preprint arXiv:1603.06270*, 2016. 7

[39] J. Zhao, M. Mathieu, R. Goroshin, and Y. Lecun. Stacked what-where auto-encoders. *arXiv preprint arXiv:1506.02351*, 2015. 1, 2