Joint autoencoders: a flexible meta-learning framework

Baruch Epstein, Ron Meir, and Tomer Michaeli

Viterbi Faculty of Electrical Engineering, Technion - Israel Institute of Technology [*]
baruch.epstein@gmail.com, rmeir@ee.technion.ac.il, tomer.m@ee.technion.ac.il

Abstract. We develop a framework for learning multiple tasks simultaneously, based on sharing features that are common to all tasks, achieved through the use of a modular deep feedforward neural network consisting of shared branches, dealing with the common features of all tasks, and private branches, learning the specific unique aspects of each task. Once an appropriate weight sharing architecture has been established, learning takes place through standard algorithms for feedforward networks, e.g., stochastic gradient descent and its variations. The method deals with meta-learning (such as domain adaptation, transfer and multi-task learning) in a unified fashion, and can deal with data arising from different modalities. Numerical experiments demonstrate the effectiveness of learning in domain adaptation and transfer learning setups, and provide evidence for the flexible and task-oriented representations arising in the network. In particular, we handle transfer learning between multiple tasks in a straightforward manner, as opposed to many competing state-of-the-art methods, that are unable to handle more than two tasks. We also illustrate the network's ability to distill task-specific and shared features.

Keywords: Autoencoders · meta-learning · weakly-supervised learning

1 Introduction

A major goal of inductive learning is the selection of a rule that generalizes well based on a finite set of examples. It is well-known [**?**], and quantified rigorously in precise terms (e.g., chapter 7 in [**?**]), that inductive learning is impossible unless some regularity assumptions are made about the world. Such assumptions, by their nature, go beyond the data, and are based on prior knowledge achieved through previous interactions with 'similar' problems. Following its early origins [**?**,**?**], the incorporation of prior knowledge into learning has become a major effort recently, and is gaining increasing success by relying on the rich representational flexibility available through current deep learning schemes [**?**]. Various aspects of prior knowledge are captured in different settings of meta-learning,

[*] This work was supported by the Technion's Skillman Chair in Biomedical Sciences and by the Ollendorff Center of the Viterbi Faculty of Electrical Engineering, Technion.

such as learning-to-learn, domain adaptation, transfer learning, multi-task learning, etc. (e.g., [?]). In this work, we consider the setup of multi-task learning, first formalized in [?], where a set of tasks is available for learning, and the objective is to extract knowledge from a subset of tasks in order to facilitate learning of other, related, tasks. Within the framework of representation learning, the core idea is that of shared representations, allowing a given task to benefit from what has been learned from other tasks, since the shared aspects of the representation are based on more information ([?]).

We consider both unsupervised and semi-supervised learning setups. In the former setting we have several related datasets, arising from possibly different domains, and aim to compress each dataset based on features that are shared between the datasets, and on features that are unique to each problem. Neither the shared nor the individual features are given apriori, but are learned using a deep neural network architecture within an autoencoding scheme. While such a joint representation could, in principle, serve as a basis for supervised learning, it has become increasingly evident that representations should contain some information about the output (label) identity in order to perform well, and that using pre-training based on unlabeled data is not always advantageous (e.g., chap. 15 in [?]). However, since unlabeled data is far more abundant than labeled data, much useful information can be gained from it. We therefore propose a joint encoding-classification scheme where both labeled and unlabeled data are used for the multiple tasks, so that internal representations found reflect both types of data, but are learned simultaneously.

The main contributions of this work are: *(i)* A generic and flexible **modular setup** for combining unsupervised, supervised and transfer learning. *(ii)* Efficient **end-to-end transfer learning** using mostly unsupervised data (i.e., very few labeled examples are required for successful transfer learning), capable of **handling multiple tasks** with ease. *(iii)* Explicit extraction of **task-specific** and **shared representations**.

2 Related work

Previous related work can be broadly separated into two classes of models: *(i)* Generative models attempting to learn the input representations. *(ii)* Non-generative methods that construct separate or shared representations in a bottom-up fashion driven by the inputs.

We first discuss several works within the non-generative setting. The Deep Domain Confusion (DDC) algorithm in [?] studies the problems of unsupervised domain adaptation based on sets of unlabeled samples from the source and target domains, and supervised domain adaptation where a (usually small) subset of the target domain is labeled. By incorporating an adaptation layer and a domain confusion loss they learn a representation that optimizes both classification accuracy and domain invariance, where the latter is achieved by minimizing an appropriate discrepancy measure. By maintaining a small distance between the source and target representations, the classifier makes good use of the relevant prior knowledge. The algorithm suggested in [?] augments standard deep learning with a domain classifier that is connected to the feature extractor, and acts to modify the gradient during backpropagation. This adaptation promotes the similarity between the feature distributions in a domain adaptation task. The Deep Reconstruction Classification Network (DRCN) in [?] tackles the unsupervised domain adaptation task by jointly learning a shared encoding representation of the source and target domains based on minimizing a loss function that balances between the classification loss of the (labeled) source data and the reconstruction cost of the target data. The shared encoding parameters allow the target representation to benefit from the ample source supervised data. In addition to these mostly algorithmic approaches, a number of theoretical papers have attempted to provide a deeper understanding of the benefits available within this setting [?,?].

Next, we mention some recent work within the generative approach, briefly. Recent work has suggested several extensions of the increasingly popular Generative Adversarial Networks (GAN) framework [?]. GANs aim to learn generative models of the data distribution by mapping specified latent distributions to highly complex data distributions. This basic approach is formulated as a minimax game, based on a generator network that aims to construct the data distribution from arbitrary latent distributions, while a discriminator tries to distinguish between the generated and the true distributions. The Coupled Generative Adversarial Network (CoGAN) framework in [?] aims to generate pairs of corresponding representations from inputs arising from different domains. They propose learning joint distributions over two domains based only on samples from the marginals. This yields good results for small datasets, but is unfortunately challenging to achieve for large adaptation tasks, and is computationally cumbersome. The Adversarial Discriminative Domain Adaptation (ADDA) approach [?] subsumes some previous results within the GAN framework of domain adaptation. The approach learns a discriminative representation using the data in the labeled source domain, and then learns to adapt the model for use in the (unlabeled) target domain through a domain adversarial loss function. The idea is implemented through a minimax formulation similar to the original GAN setup. Other relevant work in this direction includes [?,?] and [?].

The extraction of shared and task-specific representations is the subject of a number of works, such as [?], [?] and [?]. However, works in this direction typically require inputs of the same dimension and for the sizes of their shared and task-specific features to be the same. A more flexible approach can be seen in [?], proposing to augment the feature spaces of the source and target problems. Namely, the augmented source (target) version of the data includes general and source-specific (target-specific) features. In a sense, our work can be considered an extension of this strategy. However, [?] has no unsupervised learning component, does not discuss learning multiple tasks simultaneously (as opposed to transfer learning), and concerns itself with experiments from the language processing domain, solely.

A great deal of work has been devoted to multi-modal learning where the inputs arise from different modalities. Exploiting data from multiple sources (or views) to extract meaningful features, is often done by seeking representations that are sensitive only to the common variability in the views and are indifferent to view-specific variations. Many methods in this category attempt to maximize the correlation between the learned representations, as in the linear canonical correlation analysis (CCA) technique and its various nonlinear extensions [?,?]. Other methods use losses based on both correlation and reconstruction error in an auto-encoding like scheme [?], or employ diffusion processes to reveal the common underlying manifold [?]. However, all multi-view representation learning algorithms rely on *paired examples* from the two views. This setting is thus very different from transfer learning, multi-task learning, or domain adaptation, where one has access only to *unpaired samples* from each of the domains.

While GANs provide a powerful approach to multi-task learning and domain adaptation, they are often hard to train and fine tune [?]. Our approach offers a complementary non-generative perspective, and operates in an end-to-end fashion allowing the parallel training of multiple tasks, incorporating both unsupervised, supervised and transfer settings within a single architecture. This simplicity allows the utilization of standard optimization techniques for regular deep feedforward networks, so that any advances in that domain translate directly into improvements in our results. The approach does not require paired inputs and can operate with inputs arising from entirely different domains, such as speech and audio (although this has not been demonstrated empirically here). Our work is closest to [?] which shares with us the separation into common and private branches as well as unsupervised learning. They base their optimization on several loss functions beyond the reconstruction and classification losses, enforcing constraints on intermediate representations. Specifically, they penalize differences between the common and private branches of the same task, and encourage similarity between the different representations of the source and target in the common branch. This multiplicity of loss functions adds several free parameters to the problem that require further fine-tuning. Our framework uses only losses penalizing reconstruction and classification errors, thereby directly focusing on the task without adding internal constrains. Moreover, since DSN does not use a classification error for the target it cannot use labeled targets, and thus can only perform unsupervised transfer learning. Also, due to the internal loss functions, it is not clear how to extend DSN to multi-task learning, which is immediate in our formalism. Practically, the proposed DSN architecture is costly; it is larger by more than on order of magnitude than either the models we have studied or ADDA. Thus it is computationally challenging as well as relatively struggling to deal with small datasets.

3 Joint autoencoders

In this section, we introduce *joint autoencoders* (JAE), a general method for multi-task learning by unsupervised extraction of features shared by the tasks as well as features specific to each task. We begin by presenting a simple case, point out the various possible generalizations, and finally describe two transfer and multi-task learning procedures utilizing joint autoencoders.

3.1 Joint autoencoders for reconstruction

Consider a multi-task learning scenario with T tasks $t^1, ..., t^T$ defined by domains $\left\{\left(\mathcal{X}^i\right)\right\}_{i=1}^T$. Each task t^i is equipped with a set of unlabeled samples $\left\{x_n^i \in \mathcal{X}^i\right\}_{n=1}^{N^{i,u}}$,where $N^{i,u}$ denotes the size of the unlabeled data set, and with a reconstruction loss function $\ell_r^i\left(x_n^i, \tilde{x}_n^i\right)$, where $\tilde{x}_n^i$ is the reconstruction of the sample x_n^i. Throughout the paper, we will interpret ℓ_r^i as the L_2 distance between x_n^i and $\tilde{x}_n^i$, but in principle ℓ_r^i can represent any unsupervised learning goal. The tasks are assumed to be related, and we are interested in exploiting this similarity to improve the reconstruction. To do this, we make the following two observations:

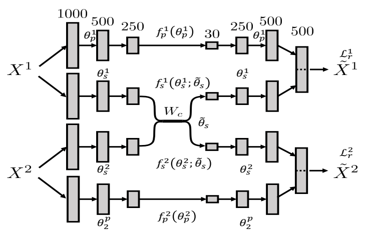
(i) Certain aspects of the unsupervised tasks we are facing may be similar, but other aspects may be quite different (e.g., when two domains contain color and grayscale images, respectively).

(ii) The similarity between the tasks can be rather "deep". For example, cartoon images and natural images may benefit from different low-level features, but may certainly share high-level structures. To accommodate these two observations, we associate with each task t^i a pair of functions: $f_p^i\left(x; \theta_p^i\right)$, the "private branch", and $f_s^i\left(x; \theta_s^i, \tilde{\theta}_s\right)$, the "shared branch". The functions f_p^i are responsible for the task-specific representations of t^i and are parametrized by parameters θ_p^i. The functions f_s^i are responsible for the shared representations, and are parametrized, in addition to parameters θ_s^i, by $\tilde{\theta}_s$ shared by all tasks. The key idea is that the weight sharing forces the common branches to learn to represent the common features of the two sources. Consequently, the private branches are implicitly forced to capture only the features that are not common to the other task. We aim at minimizing the cumulative weighted loss

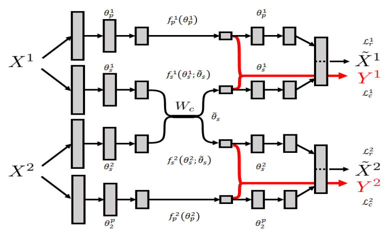
$$\mathcal{L}_r = \sum_{i=1}^{T} w_r^i \sum_{n=1}^{N^{i,u}} \ell_r^i\left(x_n^i, f_p^i\left(x_n^i; \theta_p^i\right), f_s^i\left(x_n^i; \theta_s^i, \tilde{\theta}_s\right)\right). \quad (1)$$

In practice, we implement all functions as autoencoders and the shared parameters $\tilde{\theta}_s$ as the bottleneck of the shared branch of each task, with identical weights across the tasks. Our framework, however, supports more flexible sharing as well, such as sharing more than a single layer, or even partially shared layers. The resulting network can be trained with standard backpropagation on all reconstruction losses simultaneously. Figure 1(a) illustrates a typical autoencoder for the MNIST dataset [**?**], and Figure 1(b) illustrates the architecture obtained from implementing all branches in the formal description above with such autoencoders (AE). We call this architecture a *joint autoencoder* (JAE).

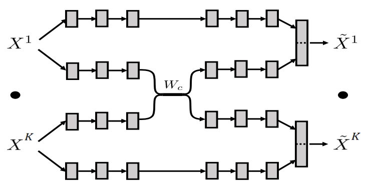
(a)



(b)



(c)



(d)

Fig. 1: (a) An example of an MNIST autoencoder (b) The joint autoencoder constructed out of the AE in (a), where $X^1 = \{0,1,2,3,4\}$ and $X^2 = \{5,6,7,8,9\}$. Each layer is a fully connected one, of the specified size, with ReLU activations. The weights shared by the two parts are denoted by W_c. The pairs of the top fully connected layers of dimension 500 are concatenated to create a layer of dimension 1000 which is then used directly to reconstruct the input of size 784. (c) A schematic depiction of a JAE architecture extended for supervised learning. The parameters and functions in figures (b) and (c) are explained in the main text. (d) A schematic depiction of a JAE architecture for K datasets. Each dataset is associated with a 'private' branch, as well as a 'shared' branch that includes the weights W_c shared by all tasks. It is also possible to design JAEs with multiple tied weights W_c^S, each shared by a subset $S \subset K$ of the datasets.

As mentioned before, in this simple example, both inputs are MNIST digits, all branches have the same architecture, and the bottlenecks are single layers of the same dimension. However, this need not be the case. The inputs can be entirely different (e.g., image and text), all branches may have different architectures, the bottleneck sizes can vary, and more than a single layer can be shared. Furthermore, the shared layers need not be the bottlenecks, in general. Finally, the generalization to more than two tasks is straightforward - we simply add a pair of autoencoders for each task, and share some of the layers of the common-feature autoencoders (Figure 1(d)). Weight sharing can take place between subsets of tasks, and can occur at different levels for the different tasks.

3.2 Joint autoencoders for multi-task, semi-supervised and transfer learning

Consider now a situation in which, in addition to the unlabeled samples from all domains $\mathcal{X}^i$, we also have datasets of labeled pairs $\left\{\left(x_k^i, y_k^i\right)\right\}_{k=1}^{N^{i,l}}$ where $N^{i,l}$ is the size of the labeled set for task t^i, assumed to be much smaller than $N^{i,u}$. The supervised component of each task t^i is reflected in the supervised loss $\ell_c^i\left(y_n^i, \tilde{y}_n^i\right)$, typically multi-class classification. We extend our loss definition in Equation 1 to be

$$\mathcal{L} = \mathcal{L}_r + \mathcal{L}_c = \mathcal{L}_r + \sum_{i=1}^{T} w_c^i \sum_{n=1}^{N^{i,l}} \ell_c^i \left(y_n^i, f_p^i\left(x_n^i; \theta_p^i\right), f_s^i\left(x_n^i; \theta_s^i, \tilde{\theta}_s\right)\right), \qquad (2)$$

where we now interpret the functions f_s^i,f_p^i to also output a classification. Figure 1(c) illustrates the schematic structure of a JAE extended to include supervised losses. Note that this framework supports various learning scenarios. Indeed, if a subset of the tasks has $N^{i,l} = 0$, the problem becomes one of unsupervised domain adaptation. The case where $N^{i,l}$ are all or mostly small describes semi-supervised learning. If some of the labeled sets are large while the others are either small or empty, we find ourselves facing a transfer learning challenge. Finally, when all labeled sets are of comparable sizes, this is multi-task learning, either supervised (when $N^{i,l}$ are all positive) or unsupervised (when $N^{i,l} = 0$).

We describe two strategies to improve supervised learning by exploiting shared features.

Common-branch transfer In this approach, we first train joint autoencoders on both source and target tasks simultaneously, using all available unlabeled data. Then, for the source tasks (the ones with more labeled examples), we fine-tune the branches up to the shared layer using the sets of labeled samples, and freeze the learned shared layers. Finally, for the target tasks, we use the available labeled data to train only its private branches while fixing the shared layers fine-tuned on the source data (see the supplementary material for a quantitative comparison to the alternative).

End-to-end learning The second, *end-to-end* approach, combines supervised and unsupervised training. Here we extend the JAE architecture by adding new layers, with supervised loss functions for each task; see Figure 1(c). We train the new network using all losses from all tasks simultaneously - reconstruction losses using unlabeled data, and supervised losses using labeled data. When the size of the labeled sets is highly non-uniform, the network is naturally suitable for transfer learning. When the labeled sample sizes are roughly of the same order of magnitude, the setup is suitable for semi-supervised learning. We find that this end-to-end strategy is not only simpler but also outperforms the alternative consistently, and use it in all subsequent experiments.

3.3 On the depth of sharing

It is common knowledge that similar *low-level features* are often helpful for similar tasks. For example, in many vision applications, CNNs exhibit the same Gabor-type filters in their first layer, regardless of the objects they are trained to classify. This observation makes low-level features immediate candidates for sharing in multi-task learning settings. However, unsurprisingly, sharing low-level features is not as beneficial when working with domains of different nature (e.g., handwritten digits vs. street signs). Our approach allows to share weights in deeper layers of a neural net, while leaving the shallow layers un-linked. The key idea is that by forcing all shared-branch nets to share deep weights, their preceding shallow layers must learn to transform the data from the different domains into a common form. We support this intuition through several experiments. As our preliminary results in Section 4.2 show, for similar domains, sharing deep layers provides the same performance boost as sharing shallow layers. Thus, we pay no price for relying only on "deep similarities". But for domains of a different nature, sharing deep layers has a clear advantage.

4 Experiments

All experiments were implemented in Keras [?] over Tensorflow [?]. The code will be made available on publication, and the network architectures and training parameters used are given in detail in the supplementary material.

4.1 Unsupervised learning

We present experimental results demonstrating the improvement in unsupervised learning of multiple tasks on the MNIST, CIFAR-10 [?] and celebA [?] datasets. For CIFAR-10 we trained the baseline autoencoder on single-class subsets of the database (e.g., all airplane images) and trained the JAE on pairs of such subsets. Table 1 shows a few typical results, demonstrating a consistent advantage for JAEs. Besides the lower reconstruction error, we can see that visually similar image classes, enjoy a greater boost in performance. For instance, the pair deer-horses enjoyed a performance boost of 37%, greater than the typical boost of $33-35\%$. The autoencoders had the same cumulative bottleneck size as the JAE, to ensure the same hidden representation size. To ensure we did not benefit solely from increased capacity, we also compared the AEs to a JAE with the same total number of parameters as the baseline, obtained by reducing the size of each layer by $\sqrt{2}$. We achieved a $22-24\%$ boost, retaining most of the advantage over the baseline. Thus, the observed improvement is clearly not a result of mere increased network capacity.

For the experiments with the celebA dataset, a collection of faces with diverse attributes, we collected $12,000$ faces of men and women each, and trained a JAE on both these subsets. The training sets contained $10,000$ samples, while the test sets contained the remaining 2000 samples. When enforcing the same number of parameters in the JAE and the pair of baseline AEs, we observed a performance boost of 24%, consistent with our findings from CIFAR-10.

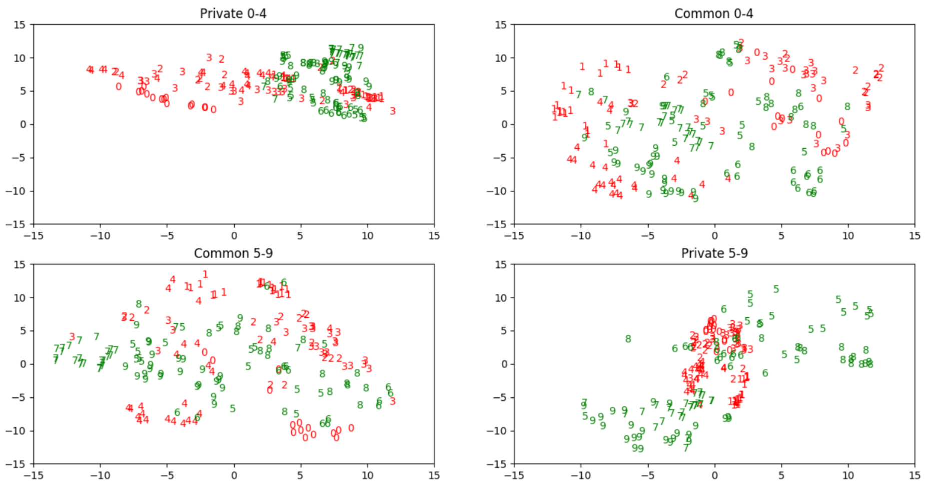
For the MNIST experiment, we have separated the training images into two subsets: X^1, containing the digits $\{0-4\}$ and X^2, containing the digits $\{5-9\}$. We compared the L_2 reconstruction error achieved by the JAE to a baseline of a pair of AEs trained on each dataset with architecture identical to a single branch of the JAE. The joint autoencoder (MSE =5.4) out-performed the baseline (MSE = 5.6) by 4%. The autoencoders had the same cumulative bottleneck size as the JAE, to ensure the same hidden representation size. As with CIFAR-10, we also compared the pair of autoencoders to a JAE with the same total number of parameters (obtained by $\sqrt{2}$ size reduction of each layer). This model achieved an MSE of 5.52, a 1.4% improvement over the baseline despite the simplicity of each single task.

Table 1: JAE reconstruction performance for CIFAR-10

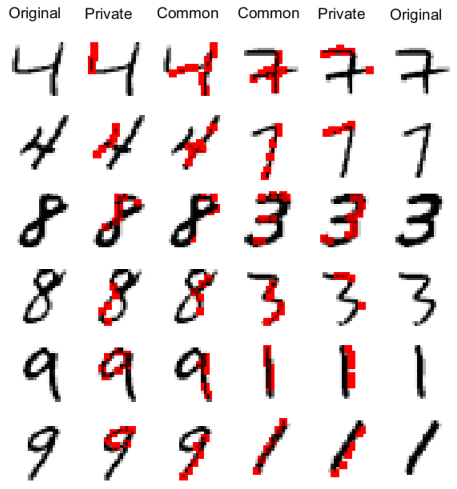
	A-D	A-H	A-S	D-H	D-S	H-S
AE error	20.8	18.5	16.2	20.6	18.2	16.0
JAE error	13.9	12.2	10.8	13.2	11.4	10.6
JAE-reduced error	16.2	14.2	12.6	15.6	14.0	12.3
JAE Improvement	33%	34%	33%	37%	35%	34%
JAE-reduced Improvement	22%	23%	22%	24%	23%	23%

Performance of JAEs and JAEs reduced by a $\sqrt{2}$ factor vs standard AEs in terms of reconstruction MSE on pairs of objects in CIFAR-10: airplanes (A), deer (D), horses (H), ships(S). For each pair of objects, we give the mean AE error, JAE and JAE-reduced error and the improvement percentage.

To further understand the features learned by the shared and private branches, we have visualized the activations of the bottlenecks and final reconstructions in the MNIST experiment. First, we considered pairs of typical digits sharing common geometric structure, such as $\{1,9\}$, $\{3,8\}$ and $\{4,7\}$. Figure 2(b) highlights the pixels in each such digit that correspond to peaks in the activity of the common and private bottleneck layers. We see that the private bottleneck focuses on the elements specific to each digit, such as the circle in $'9'$, and the left side curves in $'8'$, absent in $'1'$ and $'3'$, respectively. In contrast, the common features are sensitive to elements shared by both digits, such as vertical line in$\{1,9\}$ and the vertical-line-with-horizontal-branch pattern in $\{4,7\}$. Next, we present the $2D$ t-SNE embeddings [**?**] of the responses of each bottleneck to the digits $0-9$. The digits in first half, $\{0-4\}$, are plotted in red, while the digits in $\{5-9\}$, are plotted in green. We expect the private branch dedicated to the reconstruction of



(a)



(b)

(c)

Fig. 2: MNIST visualizations: (a) t-SNE visualizations of the responses of each bottleneck to images from $\{0 - 4\}$ (red) and $\{5 - 9\}$ (green) MNIST digits. (b) Pairs of digits sharing geometric features, with the areas corresponding to bottleneck activation peaks highlighted in red. (c) From left to right: original digits, reconstruction by the JAE, reconstruction by the private branch, reconstruction by the shared branch.

$\{0-4\}$ to map the instances of these digits to distinct clusters, and to map the digits it was never exposed to, $\{5-9\}$, to a dense small cluster, as this branch never learned to extract meaningful information about them and thus should treat them almost as noise. The same goes for the private branch dedicated to the reconstruction of $\{5-9\}$. In contrast, if the extraction of shared features took place, then the shared branches can be expected to extract information about both datasets, even those digits the branches were not dedicated to reconstruct, through the shared weights. Indeed, we see in Figure 2(a) that our expectations are met. We verify quantitatively the claim about the differences in separation between the private and shared branches. The Fisher criterion [**?**] for the separation between the t-SNE embeddings of the private branches is $9.14 \cdot 10^{-4}$, whereas its counterpart for the shared branches is $2.77 \cdot 10^{-4}$, 3.3 times less. Moreover, the shared branch embedding variance for both datasets is approximately identical, whereas the private branches map the dataset they were trained on to locations with variance greater by 2.12 than the dataset they had no access to. This illustrates the extent to which the shared branches learn to separate both datasets better than the private ones.

Finally, Figure 2(c) displays examples of digits reconstructions. The columns show (from left to right) the original digit, the image reconstructed by the full JAE, the output of the private branches and the shared branches. We see that the common branches capture the general shape of the digit, while the private branches capture the fine details which are specific to each subset.

We remark that we experimented with an extension of unsupervised JAEs to variational autoencoders [**?**]. Unlike standard VAEs, we trained three hidden code layers, requiring each to have a small Kullback-Leibler divergence from a given normal distribution. One of these layers was used to reconstruct both datasets (analogous to the shared bottleneck in a JAE), while the other two were dedicated each to one of the datasets (analogous to the private branches). The reconstruction results on the halves of the MNIST dataset were promising, yielding an improvement of 12% over a pair of VAEs of the same cumulative size. Unfortunately, we were not able to achieve similar results on the other datasets, nor to perform efficient multi-task\ transfer learning with joint VAEs. This remains an intriguing project for the future.

4.2 Transfer learning

Next, we compare the performance on MNIST of the two JAE-based transfer learning methods detailed in Section 3.2. For both methods, X^1 contains digits from $\{0-4\}$ and X^2 contains the digits $\{5-9\}$. The source and target datasets comprise 2000 and 500 samples, respectively. All results are measured on the full MNIST test set. The common-branch transfer method yields 92.3% and 96.1% classification precision for the $X^1 \to X^2$ and $X^2 \to X^1$ transfer tasks, respectively. The end-to-end approach results in 96.6% and 98.3% scores on the same tasks, which demonstrates the superiority of the end-to-end approach.

Shared layer depth We investigate the influence of shared layer depth on the transfer performance. We see in Table 2 that for highly similar pairs of tasks such as the two halves of the MNIST dataset, the depth has little significance, while for dissimilar pairs such as MNIST-USPS, "deeper is better" - the performance improves with the shared layer depth. We believe that if the inputs are very different, the lower layers transform them into a similar representation using very different features, and then use the shared deep layers to perform the transfer learning (see discussion in 3.3). Moreover, when the input dimensions differ, early sharing is impossible - the data must first be transformed to have the same dimensions.

Table 2: Shared Layer Depth and Transferability

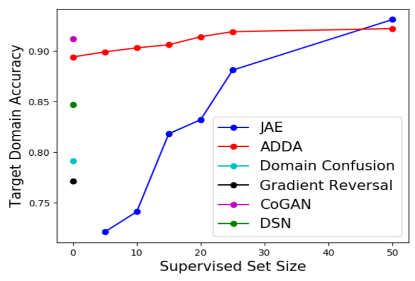
	1	2	3	4	5
MNIST {0-4} → {5-9}	96.5	95.4	95.8	96.1	96.0
MNIST {5-9} → {0-4}	98.3	97.6	97.8	98.2	98.3
MNIST → USPS			84.8		87.6
USPS → MNIST			83.2		86.9

Influence of the shared layer depth on the transfer learning performance (target test accuracy). For the MNIST-USPS pair, only partial data are available for dimensional reasons.

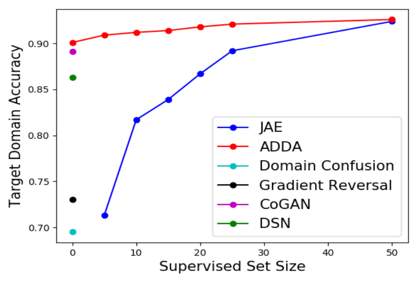
MNIST, USPS and SVHN digits datasets We have seen that the end-to-end JAE-with-transfer algorithm outperforms the alternative approach. We now compare it to other domain adaptation methods that use little to no target samples for supervised learning, applied to the MNIST, USPS and SVHN digits datasets. The transfer tasks we consider are MNIST→USPS , USPS→MNIST and SVHN→MNIST. Following [**?**] and [**?**], we use 2000 samples for MNIST and 1800 samples from USPS. For SVHN→MNIST, we use the complete training sets. In all three tasks, both the source and the target samples are used for the unsupervised JAE training. In addition, the source samples are used for the source supervised element of the network. We study the weakly-supervised performance of JAE and ADDA allowing access to a **small number of target samples**, ranging from 5 to 50 per digit. For the supervised version of ADDA, we fine-tune the classifiers using the small labeled target sets after the domain adaptation. Figure 3 $(a)-(c)$ provides the results of our experiments. For recent methods such as CoGAN, gradient reversal, domain confusion and DSN, we display results with zero supervision, as these methods do not support weakly-supervised training. For DSN, we provide preliminary results on MNIST↔USPS, without model optimization that is likely to prevent over-fitting.

On all tasks, we achieve results comparable or superior to existing methods using very limited supervision, despite JAE being both conceptually and computationally simpler than competing approaches. In particular, we do not train a GAN as in CoGAN, and require a single end-to-end training period, unlike ADDA that trains three separate networks in three steps. Computationally, the models used for MNIST→USPS and USPS→MNIST have $1.36M$ parameters, whereas ADDA uses over $1.5M$ weights. For SVHN→MNIST, we use a model with $3M$ weights, comparable to the $1.5M$ parameters in ADDA and smaller by an order of magnitude than DSN. The SVHN→MNIST task is considered the hardest (for instance, GAN-based approaches fail to address it) yet the abundance of unsupervised training data allows us to achieve good results, relative to previous methods. We provide further demonstration that knowledge is indeed transferred from the source to the target in the MNIST→USPS transfer task with 50 samples per digit. Source supervised learning, target unsupervised learning and target classifier training are frozen after the source classifier saturates (epoch 4). The subsequent target test improvement by 2% is due solely to the source dataset reconstruction training, passed to the target via the shared bottleneck layer (Figure 3(d)).

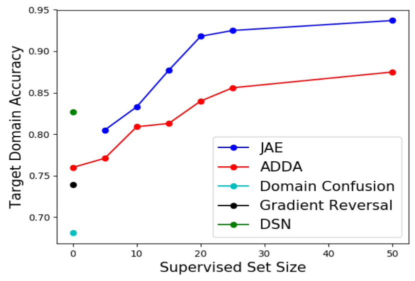
Three-way Transfer Learning We demonstrate the ability to extend our approach to multiple tasks with ease by transferring knowledge from SVHN to MNIST and USPS simultaneously. That is, we train a triple-task JAE reconstructing all three datasets, with additional supervised training on SVHN and weakly-supervised training on the target sets. All labeled samples are used for the source, while the targets use 50 **samples per digit**. The results illustrate the benefits of multi-task learning: 94.5% classification accuracy for MNIST, a 0.8% improvement over the SVHN→MNIST task, and 88.9% accuracy in UPS, a 1.2% improvement over SVHN→USPS. This is consistent with unsupervised learning being useful for the classification. USPS is much smaller, thus it has a lower score, but it benefits relatively more from the presence of the other, larger, task. We stress that the extension to multiple tasks was straightforward, and indeed we did not tweak the various models, opting instead for previously used JAEs, with a single shared bottleneck. Most state-of-the-art transfer methods do not allow for an obvious, immediate adaptation for transfer learning between multiple tasks. Indeed, [**?**] would require a number of loss functions growing quadratically in the task number, and an even more demanding architecture than is already employed, while [**?**] would require a quadratically growing amount of discriminators, or else a novel idea to perform efficient domain adaptation for multiple tasks. It is even less clear how to extend [**?**] to such scenarios.



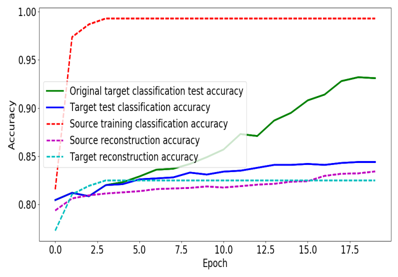
(a)



(b)



(c)



(d)

Fig. 3: Transfer learning results on MNIST, USPS and SVHN. (a) MNIST→USPS. 50 samples per digit allow JAE to surpass ADDA and CoGAN, reaching 93.1% accuracy. (b) USPS→MNIST. 25 samples per digit allow JAE to surpass CoGAN. At 50 samples per digit, JAE (92.4%) is comparable to ADDA (92.6%). (c) SVHN→MNIST. JAE achieves state-of-the-art performance, reaching 93.7% accuracy. CoGAN did not converge on this task. (d) Transfer occurring purely due to source unsupervised learning. The green graph is the test classification accuracy achieved without our interference by freezing the source supervised learning, target unsupervised learning and target classifier training. The reconstruction accuracy is measured as the fraction of pixels correctly classified as white\black. Equivalently, it is the complement of the average reconstruction error.

5 Conclusion

We presented a general scheme for incorporating prior knowledge within deep feedforward neural networks for domain adaptation, multi-task and transfer learning problems. The approach is general and flexible, operates in an end-to-end setting, and enables the system to self-organize to solve tasks based on prior or concomitant exposure to similar tasks, requiring standard gradient based optimization for learning. The basic idea of the approach is the sharing of representations for aspects which are common to all domains/tasks while maintaining private branches for task-specific features. The method is, in principle, applicable to data from multiple sources and types, though adapting it for non-image domains is an ongoing research project. It also has the advantage of being able to share weights at arbitrary network levels, enabling abstract levels of sharing.

We demonstrated the efficacy of our approach on several domain adaptation and transfer learning problems, and provided intuition about the meaning of the representations in various branches. In a broader context, it is well known that the imposition of structural constraints on neural networks, usually based on prior domain knowledge, can significantly enhance their performance. The prime example of this is, of course, the convolutional neural network. Our work can be viewed within that general philosophy, showing that improved functionality can be attained by the modular prior structures imposed on the system, while maintaining simple learning rules.