IEEE Robotics and Automation Letters (RAL) paper presented at the
2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)
October 25-29, 2020, Las Vegas, NV, USA (Virtual)

Explicit Domain Adaptation with Loosely Coupled Samples

Oliver Scheel[1,2], Loren Schwarz[1], Nassir Navab[2], Federico Tombari[2,3]

***Abstract*— Transfer learning is an important field of machine learning in general, and particularly in the context of fully autonomous driving, which needs to be solved simultaneously for many different domains, such as changing weather conditions and country-specific driving behaviors. Traditional transfer learning methods often focus on image data and are black-box models. In this work we propose a transfer learning framework, core of which is learning an explicit mapping between domains. Due to its interpretability, this is beneficial for safety-critical applications, like autonomous driving. We show its general applicability by considering image classification problems and then move on to time-series data, particularly predicting lane changes. In our evaluation we adapt a pre-trained model to a dataset exhibiting different driving and sensory characteristics.**

I. INTRODUCTION

Transfer learning aims to apply models to domains different from those they were originally trained on - possibly even to different tasks - while leveraging existing knowledge. This process, which is mastered exceptionally well by humans, is an interesting and important field of research, and an important step towards the concept of general artificial intelligence. Further, it mitigates some challenges of current deep learning algorithms, such as the need to collect and annotate huge amounts of data for each domain and task.

In the field of autonomous driving, transfer learning also plays a crucial role, but is among the lesser explored topics. Here, many domains are conceivable and necessary, and models need to be able to cope with a variety of complex environments and situations: a camera system needs to produce reliable detections under varying weather and illumination conditions - a trajectory prediction model has to function for rural areas and crowded megacities, considering location- and region-specific driving behaviors (see Figure 1 as an example). Other changes stem from within the vehicle itself: when sensor configurations or software versions change, do neural networks from previous versions have to be trained from scratch? Furthermore, model transfer is strongly coupled with the problem of safety: having demonstrated that a model satisfies certain functional safety requirements, can we move to new domains so that these guarantees still hold?

This gives rise to the need for explainable transfer learning techniques. In this work we propose such a method, learning an explicit, linear mapping between domains in the form of a transformation matrix. Related work targets image-to-image translation and style transfer, often using adversarial approaches [1], which are not interpretable and further focus only on image data. Although explicit domain adaptation algorithms have been proposed, often they are specific to certain tasks and models [2], [3], whereas we introduce a more general framework with a novel correspondence loss.

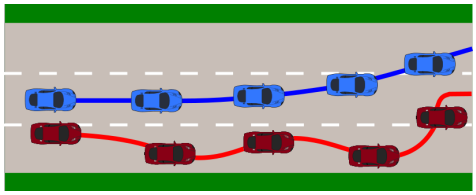

Fig. 1. Comparison of two lane changes in different domains. The blue car executes its lane change smoothly, while the red one exhibits a noisy driving style, causing many false predictions in models not exposed to this.

Core of our proposed method is learning a mapping, with which samples can be converted between domains. Advantages of such an explicit transformation are several, e.g. the previously mentioned interpretability but also the possibility of integrating domain knowledge. Due to characteristics of autonomous driving problems and available data, main focus of our work are high-level, numerical and sequential representations. We motivate relevant applications on toy tasks, and address the real-world problem of predicting lane changes, for which we tackle the challenging problem of transferring across varying sensor setups and driving styles, achieving promising results. Further, we address image problems, as well, particularly digit classification on MNIST, showing the general applicability of our proposed framework.

II. RELATED WORK

One common transfer learning technique is fine-tuning of pre-trained networks, for which lower levels of the network are frozen, while only layers close to the output are trained on the new domain or task. This often outperforms training from scratch, due to the amount of information already stored in the network, and leads to state-of-the-art algorithms, even for totally different tasks [4], [5].

Yim et al. extend the concept of knowledge distillation [6] to include transfer learning applications with a focus on network compression [7].

Domain adaptation methods have been frequently applied in the field of image translation, e.g. for transferring image styles or creating synthetic training images in different domains [8], [9]. Zhu et al. achieve fascinating results for unpaired image-to-image translation, using two Generative Adversarial Networks (GANs) and a cycle consistency loss [1]. In contrast to this, in our method we make use of existing (loosely) coupled training samples, and calculate an explicit

[1]BMW Group, München, Germany `first.lastname@bmw.de`
[2]Faculty of Computer Science, Technische Universität München, Garching bei München, Germany
[3]Google Inc., Switzerland

transformation matrix $\mathbf{T}$. This simplifies understanding of the process, and is closer to the underlying idea, transforming samples between domains instead of hallucinating them. Since $\mathbf{T}$ nearly always is invertible, we are given the cyclic conversion "for free" and save parameters. Isola et al. employ a similar principle as [1], except using exact correspondence pairs (x, y) and only one generator and discriminator [10]. The generator is conditioned on input x from domain A, and tries to create the corresponding image y from domain B, s.t. the discriminator cannot distinguish from real samples of these domains. Li et al. employ linear transformation matrices for the problem of style transfer, achieving state-of-the-art results [11]. In comparison to their work, which calculates a style loss at different layers in the network, we use a correspondence loss in the actual domains, making transformations better understandable and also enabling usage in shallower architectures, like Recurrent Neural Networks (RNNs). Jaderberg et al. address the weakness of classical Convolutional Neural Networks (CNNs) and introduce Spatial Transformer Networks (STNs) to deal with transformed images [12]. Their goal neither is transfer learning and they lack the correspondence loss as we introduce it, still their idea is similar.

Duan et al. introduce transformation matrices to project data of different domains in a common space, and include these in Support Vector Machines (SVMs) [13]. Paaßen et al. also extend a common optimization concept with an explicit transformation matrix $\mathbf{H}$ [14]. While their transformation $\mathbf{H}$ is global, we calculate a transformation $\mathbf{T}_x$ for every datapoint x, which allows for greater flexibility. Paaßen only include $\mathbf{H}$ in the loss formulation, thus using a very similar loss to [12]. Aswolinskyi et al. extend [13] to unsupervised applications, learning a predictive model to generate synthetic data for domain B in an autoregressive fashion [2].

Sun et al. introduce a preprocessing procedure, dubbed CORAL, to address domain shift and leverage transfer between domains [15]. They propose, in addition to the common standardization step of modifying data to possess 0 mean and unit variance, to further minimize distances in second-order statistics (covariance) between both analyzed datasets.

Triplet loss is a common method for defining similarities of data points [16], and thus related to our interpretation of correspondence. For more details we refer to Section III-A.

III. PROBLEM DEFINITION

In all our experiments we use a source domain A and target domain B, on which we solve the same task. We have a (complex) model $\mathbf{M}$ trained on A and aim to leverage this knowledge to domain B. We assume, that data for A is plenty and sufficient, while for B it is more rare, which is a realistic application scenario. To simulate this and further analyze the effect of data quantity on transfer success, we limit domain B to b data samples for varying b.

In unison with the general idea of transfer learning, it is not desired to simply train $\mathbf{M}$ from scratch on B, as this has several disadvantages, in general, but in particular also for the field of autonomous driving: it is not desirable to train and store full, complex models for each domain (e.g. for each existing country), due to time and resource constraints. Further, any behavioral analysis of $\mathbf{M}$, e.g. for safety verification reasons - like running a simulation, had to be done anew. Additionally, for small b our experiments show bad results when training full models, among others due to the disproportion of available training data and model parameters. This was also noted by [7]. Due to this, we aim at leveraging and incorporating existing knowledge, and to build on models pre-trained on A.

A. Corresponding Samples

Core of our proposed algorithm is a novel correspondence loss, for which we need corresponding samples from both domains. There are different "degrees" of correspondence though, and we briefly discuss our correspondence requirements. [1] consider completely unpaired samples. In their method, random samples from domains A and B are drawn for training, and a generator learns to generate samples from the other domain, indistinguishable for a discriminator. Conversely, [17] fine-tune classification models for the task of image segmentation. When considering the problem of mapping images to semantic segmentation maps as domain adaptation task, images x in the original domain are paired with the corresponding segmentation x'. This is (depending on the labelling quality) nearly an exact correspondence, there exists a simple mapping f s.t. $f(x) = x'$. In [1], this mapping neither is simple nor deterministic.

For our application scenario, we relax this assumption to a rough correspondence, namely $f(x) \approx x'$, guiding our model towards the correct solution by providing n correspondence samples $(x, x'_1), \ldots, (x, x'_n)$, s.t. the learned average mapping provides a good fit - thus the name "loosely coupled". For images to semantic views, this corresponds to contrasting the original image of a scene with n semantic segmentation maps, which are slightly modified, e.g. differ in the number of parked cars on a street.

For our experiments on MNIST, an image is assigned n images from the other domain with equal label. For the problem of predicting lane changes, sequences are aligned based on labels, as well: Correspondences of follow-only sequences are follow-only sequences of the other domain. If a lane change is present, correspondences are lane changes of the other domain, aligned s.t. the execution times of maneuvers match as closely as possible, which can be done automatically in linear time for each sample, as labels are known and we can iterate over all other samples for a best fit. Throughout all our experiments, we use $n = 5$. This decision is based on empirical evidence of good performance, and further denotes a realistic number of available correspondence pairs.

To conclude this section, note the relation of our interpretation of correspondence to triplet loss [16]. In the calculation of triplet loss for an anchor point a a negative (n) and positive sample (p) is used. Our correspondence samples $(x, x'_1), \ldots, (x, x'_n)$ can be understood as pairs of

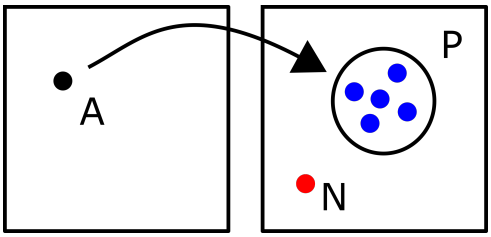

Fig. 2. Depiction of our used correspondence mapping f and its relation to triplet loss. For the triplet loss, an anchor point (A) is assigned a positive sample (P) and contrasted to a negative one (N). In our case, f, and thus the model trained with it, is a generative way of converting anchor point A from one domain to another (ideally, resulting in the mean of the positive correspondence points).

anchor point x and positive samples x'_i, and no negative samples are given. While models trained with triplet loss offer a discriminative procedure of deciding which inputs are close together, our proposed framework can be understood as a generative model for sampling such positive points (see Figure 2 for a visualization).

IV. MODEL

Core of our proposed framework is a neural network we call Converter ($\mathbf{C}$), which is responsible for converting samples from one domain to another. $\mathbf{C}$ is prepended before the actual network trained on A, and calculates a transformation matrix $\mathbf{T}_x$ for each input $x \in B$. x is then multiplied by $\mathbf{T}_x$, which equals applying the learned mapping into domain A, s.t. the resulting value resembles the corresponding input in domain A.

Formally, let x be a sample of domain B, $x \in \mathbb{R}^d$, $d \in \mathbb{N}$, with samples of A having dimensionality d'. Such arbitrary domains and domain dimensions are common in transfer learning tasks, and our framework does not require any additional constraints. However, in our experiments domains A and B are of equal dimensionality, thus for simplicity in the following we set $d = d'$. Let $\mathbf{x}$ be the homogeneous representation of x, i.e. $\mathbf{x} = \begin{pmatrix} x & 1 \end{pmatrix}^\top$. Then $\mathbf{C}$ defines the mapping

$$\mathbf{C} : B \to \mathbb{R}^{(d+1)\times(d+1)}, \; x \mapsto \mathbf{T}_x \tag{1}$$

s.t.

$$\mathbf{T}_x \mathbf{x} = \mathbf{x}' \tag{2}$$

and x' is a good representation of x in A. Assume $\mathbf{M}$ is our full model trained on A, then let $\mathbf{L}$ denote the last l layers of $\mathbf{M}$ connected to the output. $\mathbf{C}$ usually has fewer parameters than $\mathbf{M}$, as it is intended for a simpler purpose, solving part of the original task, which helps saving resources.

Training of our framework consists of three steps, each of which is optional (e.g., it can also be used without corresponding samples):

1) **Pre-training**: Initialize $\mathbf{C}$ with an educated guess of possible transformation matrices. For this, train $\mathbf{C}$ with samples $x \in B$, using some initial transformation matrix $\tilde{\mathbf{T}}_x$ as ground truth and for optimization the element-wise L2-loss $\mathfrak{L}_P(x)$ (which equals the Frobenius norm in matrix space):
$$\mathfrak{L}_P(x) = |\mathbf{C}(x) - \tilde{\mathbf{T}}_x|_F \tag{3}$$
2) **Correspondence training**: Train $\mathbf{C}$ with the correspondence pairs $(x, x'_1), \ldots (x, x'_n)$ for $x \in B$. Use a suited loss function $\mathfrak{L}_C(x)$ operable in the domain spaces, e.g. the L2-loss between converted and actual samples:
$$\mathfrak{L}_C(x) = \frac{1}{n} \sum_{i=1}^{n} |\mathbf{C}(x)\mathbf{x} - \mathbf{x}'_\mathbf{i}|_2 \tag{4}$$
where again $\mathbf{x}'_\mathbf{i}$ is the homogeneous version of x'_i.
3) **Fine-tuning**: For this, we distinguish two train modes: In mode 0, we only retrain $\mathbf{L}$ with the actual task loss (e.g. a cross-entropy loss for classification). In mode 1, we retrain $\mathbf{L}$ and $\mathbf{C}$, accumulating task loss and correspondence loss $\mathfrak{L}_C$.

Step 1 enables incorporating existing domain knowledge into the framework, as often one might have a rough estimate of how the resulting transformation could look like (e.g. a conversion between Radar and Lidar points, variation of the angle of mounted cameras ...). If such prior information is not available, it is recommendable to use $\tilde{\mathbf{T}} = \mathbf{I}^{d+1}$. This way, if the converter module cannot deduce a meaningful intra-domain transformation, the method will still be as least as good as standard fine-tuning. In Step 2 the model makes use of the given corresponding samples, in order to further refine or learn the best transformation between samples from domains A and B in a data-driven fashion. Hopefully, by now the output of $\mathbf{C}$ resembles samples of domain A, s.t. the original model $\mathbf{M}$ already exhibits good results. As such transformation is not perfect though, in Step 3 the model is fine-tuned to further increase performance. Figure 3 shows a graphical overview of our framework.

A. Simplifying Assumptions

$\mathbf{T}$ represents a transformation matrix in the domain space, and as such can be restricted to only allow certain transformations, such as rotations, affine or projective transformations. The tighter the restriction, the more limited is the model, but this simultaneously makes learning the transformation easier. In addition to this, in the following we describe simplifying adaptations for image and sequential data. Note though that in any case it is possible to apply our framework without these, working directly in the full vector spaces of all images or sequences, but that these assumptions are simplifying approximations helpful in reducing complexity.

1) Image Data: Let x be an image of size $w \times h$, thus $x \in \mathbb{R}^{w \times h}$. As image transformations are often structured (e.g. rotations, rectifying projections, ...), we apply transformations globally for the whole image, treating each pixel similarly. Instead of operating in the image space $\mathbb{R}^{w \times h}$, our transformations live in the space $\mathbb{R}^{3\times3}$, representing the common homogeneous transformations for 2D-points. We utilize the module introduced and used in STNs to

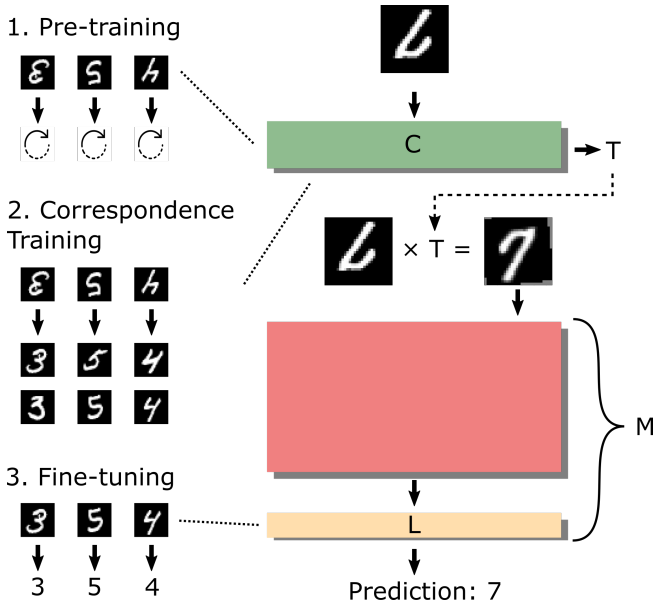

Fig. 3. Graphical overview of our framework adapting a model trained on standard MNIST to rotated images. Steps 1 and 2 consist of training the Converter $\mathbf{C}$, first pre-training it with (an expected) rotation, and then with n correspondence pairs for each sample (here $n = 2$). In Step 3, the last layers $\mathbf{L}$ of $\mathbf{M}$ are fine-tuned on the new dataset, while the complex part of $\mathbf{M}$ is frozen (drawn in red).

apply these transformations, although other approaches are possible.

2) Sequential Data: Main focus of this work is dealing with time-series data. For these we employ frame-based transformations, as well. Assume we examine sequences of length l and feature size f, thus $x \in \mathbb{R}^{f \times l}$. Instead of learning accordingly dimensioned transformations, we consider each frame separately, s.t. $\mathbf{T} \in \mathbb{R}^{(f+1)\times(f+1)}$. To model the temporal context, in this case $\mathbf{C}$ is an RNN, which expects full sequences as inputs and outputs a sequence of transformation matrices. More specifically, we use RNNs consisting of Long-Short Term Memory Units (LSTMs) [18]. We use the following shorthand notation for an LSTM cell:

$$(\mathbf{h}^t, \tilde{\mathbf{c}}^t) = \texttt{LSTM}(\mathbf{x}^t, \mathbf{h}^{t-1}, \tilde{\mathbf{c}}^{t-1})$$

where $\mathbf{x}^t$ is input at timestep t and thus t-th frame of the sequence, $\mathbf{h}$ the hidden state and $\tilde{\mathbf{c}}$ the memory unit. With weight matrix $\mathbf{W}$ and bias vector $\mathbf{b}$, the calculation of $\mathbf{T}$ in step t is given by:

$$\mathbf{T}_x^t = \texttt{softmax}(\mathbf{W} \cdot \mathbf{h}^t + \mathbf{b}). \tag{5}$$

V. DATASETS AND METRICS

In this section we introduce used datasets and metrics for evaluation. We start with two toy problems to motivate and show functionality of our proposed model, and then move to the real-world problem of predicting lane changes. For each analyzed problem we describe the architecture of the "base" model $\mathbf{M}$, which is trained on domain A and serves as starting point for the transfer task to domain B.

A. Datasets

1) Transformed MNIST: For simple motivation, we use images of the well-studied MNIST dataset [19]. While domain A are the standard images, domain B consists of images rotated by $180°$ degrees. $\mathbf{M}$ is a simple CNN consisting of 3 convolutional and max-pooling layers, followed by one fully-connected layer for classification. For fine-tuning, $\mathbf{L}$ is this last layer.

2) Toy Sequences: We simulate lane change maneuvers, representing each solely by the target car's distance to the lane's center line - ranging from 0 (at left lane border) to 1 (at right lane border). Domain A consists of "clean" lane changes, in which drivers smoothly follow their lane until the maneuver, and then also execute this in an orderly fashion. In Domain B we added noise to these trajectories, see Figure 1. This could model the realistic scenario of transferring between country-specific driving styles - one can think of countries in which traffic is more strictly regulated and rule-abiding, while in some countries trajectories are more freely chosen. One can assume, that a model trained on A and applied to B exhibits more false predictions, due to its missing exposure to noise. Indeed, this is the case, as results in Section VI show, even, when using high-level data as we do here. Naturally, the domain gap for problems concerning images or raw sensor data is greater, still, our results show a significant performance drop also for this abstracted data. $\mathbf{M}$ is similar as in the next paragraph, except using only one feature, namely distance to the lane's center line, and a smaller hidden size of 32. Further, also the problem definition including how ground truth labels are defined, is identical to the one in the next paragraph. In particular, this is a classification problem, in which each frame is given one of the labels "follow lane" or "change lanes".

3) Lane Change Prediction: For predicting lane changes, we closely follow our previous work [20], using the simple LSTM model as base model. This is an LSTM network of hidden size 64 with a single classification layer on top. For retraining, $\mathbf{L}$ is this classification layer. A multitude of features is available, for this work, though, for visualization purposes we mostly use the two features distance to lane's center line (m) and lateral velocity (v). These were proven to be the most relevant ones by [21]. Each frame is given one of the labels L(eft), F(ollow) or R(ight). The model also predicts in each step, with a prediction of L or R indicating the algorithm's belief that a lane change to the respective side is imminent. In particular, all frames of the dataset are labelled F, but frames from 3s before a lane change until its execution (the time point of the vehicle crossing lane boundaries) are labelled with the respective direction. Shortly before this time period, and after a lane change, frames are assigned weight 0, to give models time to adjust and not penalize "too early" predictions (see [20] for more details). Due to the huge imbalance in the datasets (a majority of frames is labelled F), frames are weighted inversely proportional to their class' frequency. Further, frames labelled L or R are weighted exponentially more the

closer the lane change (compare [22], [20]).

Domain A is a collection of lane changes collected by BMW in-series cars (introduced in [20]), mainly in Germany. Domain B is the publicly available NGSIM dataset [23], which is a collection of lane changes recorded on highways in the United States. Next the different recording countries, different sensor-setups, e.g. with different sampling frequencies, were used. Although these differences are diminished by the used high-level representation, among others we interpolate between frames, causing different temporal characteristics. Further, recording for NGSIM was done in busy highway sections, while recordings of the fleet data in general contain less dense traffic scenes.

B. Metrics

For evaluating lane change problems, we follow the metrics introduced in [20] and for simplicity and better comparison pick the three most relevant ones:

- **Frequency**: Number of times a lane change maneuver is falsely predicted during lane-following periods (number of false positives). A *Frequency* of 0 is optimal.
- **Delay**: delay of lane change prediction w.r.t. ground truth, measured in seconds. A delay of 0s describes a prediction, which begins as soon as the ground truth label switches to lane change (and thus predicts the maneuver 3s ahead of its execution).
- **Miss**: number of lane changes completely missed.

To better compare algorithms, we further accumulate these into one score as follows: Model $\mathbf{M}$, which was trained on A and tested on B, is used as baseline. For each subsequent algorithm, its percental increase or decrease w.r.t. this baseline in each metric is measured. The resulting score is the sum of these percentages, in which we weigh *Frequency* by the relative number of frames labelled F, and *Delay* and *Miss* by the relative number of frames labelled L or R. This way, we assign priorities to the maneuver classes according to the relative duration a driver experiences them for, and acknowledge the huge importance of *Frequency*: A controller will and has to react to each false prediction, causing driver discomfort. Furthermore, a relative decrease in *Delay* can be tolerated if in acceptable bounds, and the number of misses of our algorithms is so low, that small absolute changes here distort percentage values.

VI. RESULTS

We implement state-of-the-art baseline methods and compare our methods against these. We first introduce these in general, and in the respective sections describe any problem-specific adaptations, if existing. We would like to refer interested readers to the supplementary video, which contains additional, quantitative results, e.g. further examining interpretability of resulting transformations..

A. Baseline Methods

- Fine-tuning: We fine-tune layers $\mathbf{L}$ of model $\mathbf{M}$, which was trained on A and tested on B. Note that [7] compare against this as well, and perform marginally worse due to their focus on network compression, thus a relative comparison to their approach can be established.
- CORAL [15]: We use the CORAL algorithm to minimize covariance distances between used domains A and B, which have already been standardized. Afterwards, we fine-tune layers $\mathbf{L}$ of model $\mathbf{M}$ on the new domain.
- pix2pix [10]: To examine an implicit domain adaptation method, in contrast to our explicit one, we choose the adversarial approach from the pix2pix framework.
- Imp: To further analyze a possible trade-off between explicitness and model capabilities, this baseline equals our proposed framework, except $\mathbf{C}$ now does not output a transformation matrix $\mathbf{T}$, but directly samples from domain B. Due to this, Step 1 of our proposed training scheme is not possible and thus left out.
- Mode 2: Further, we compare against [14] and STNs [12]. Note that a slight modification of [14] (using a sample-specific transformation matrix instead of a global one) as well as STNs can be expressed by means of our framework: Both methods employ a transformation matrix, and fine-tune existing models adapting $\mathbf{C}$ and $\mathbf{L}$. They lack Steps 1 and 2, concerning pre-training and training the Converter (and lack our correspondence loss overall), but otherwise are identical to our algorithm with train mode 1, except only the classification loss is considered. We denote this small modification by Mode 2, and additionally add Step 1 with the same pre-training targets for fairness.
- Mode 0: We use similar experiments as further ablation studies, fathoming how the correspondence loss affects performance, as this is one of our contributions and a core difference to [14] and STNs: Mode 0 denotes our framework stripped of Step 2 and using train mode 0, thus excluding the correspondence loss. Both methods, Mode 2 and Mode 0, answer the question whether using such a correspondence loss is beneficial, or whether similar results can be obtained by just pre-training a converter well and eventually fine-tuning the model, including $\mathbf{C}$ or $\mathbf{L}$ only.
- Global: We test a modification of the proposed framework, using a learned global transformation instead of a sample-specific one, to examine the effects of this change, and also to better understand differences to e.g. [14]. To obtain such sample invariance, we remove the Converter's input. We only apply this approach to the MNIST example, as only here global transformations are available, and for our sequential problems transformations depend on samples and timesteps.

For clarification, we repeat explanation of train modes 0 and 1 of our proposed framework and essential differences to Mode 0 and 2: our framework consists of the three steps pre-training, correspondence training and fine-tuning. In the fine-tuning step, we distinguish between train mode 0 and 1: in the former, only $\mathbf{L}$ is retrained based on the original task loss alone, while in the latter $\mathbf{L}$ and $\mathbf{C}$ are retrained employing task and correspondence loss. Mode 0 and 2 mir-

ror methods such as STNs, which do not assume existence of correspondence pairs and thus lack the correspondence training step. Consequently, only the original task loss is (and can be) used. Mode 0 only adapts weights of $\mathbf{L}$, Mode 2 $\mathbf{L}$ and $\mathbf{C}$.

B. Rotated MNIST

Table I shows quantitative results on the MNIST dataset. The accuracy of the tested models on domain B is listed, which is artificially limited to b samples (in brackets the share of b w.r.t. the whole dataset is given). $T1$ and $T2$ denote the application of our framework while pre-training $\mathbf{T}$ with the identity and a $180°$ rotation matrix, respectively. We limit the space of possible transformations to euclidean transformations (rotations + translations). Further limiting or relaxing this assumption slightly increases or decreases performance. $\mathbf{C}$ is a simple CNN consisting of 2 convolutional and one fully-connected layer. The used train mode is 0, 1 does not improve performance. Additionally, we show results of $\mathbf{M}$ trained on A and applied to B (A on B), as well as of training all of $\mathbf{M}$ on the full dataset B (B on B) to give a theoretical upper bound of reachable performance.

The drastic drop in accuracy when switching domains shows the need for transfer learning techniques. Our framework performs at least as well or very comparably to fine-tuning. Incorporating domain knowledge ($T2$) greatly improves performance, already or especially for small b a very high classification accuracy is reached. Fine-tuning and our variant $T1$ are close together, since learning to align images via pixel-loss is difficult [24], [25]. Further, on this limited dataset the models already achieve near-optimal performance. Note that for images, since we use the same module, but also in general our approach with train mode 1, combining classification and pixel-loss, is very similar to STNs. However, since train mode 1 yields no performance increase, this shows that also STNs are not more effective in scenarios with limited training data.

TABLE I
RESULTS ON THE MNIST DATASET.

b	Model	Accuracy
100 (0.14)	Fine-tune	0.738
	CORAL	0.5
	pix2pix	0.375
	Imp	0.313
	Global - $T1$	0.688
	Global - $T2$	0.901
	Ours - $T1$	0.766
	Ours - $T2$	**0.912**
1000 (1.43)	Fine-tune	0.908
	CORAL	0.852
	pix2pix	0.898
	Imp	0.711
	Global - $T1$	0.891
	Global - $T2$	0.932
	Ours - $T1$	0.901
	Ours - $T2$	**0.947**
2000 (2.86)	Fine-tune	**0.972**
	CORAL	0.935
	pix2pix	0.859
	Imp	0.846
	Global - $T1$	0.929
	Global - $T2$	0.952
	Ours - $T1$	0.966
	Ours - $T2$	0.969
70000 (100)	A on B	0.153
	B on B	0.980

As train mode 1 shows no advantages over train mode 0 in this example, Step 2 of our training procedure is of no big use, and thus results of Mode 0 and 2 are omitted here. For this problem, the original pix2pix architecture is used. This and baseline Imp both exhibit problems dealing with this task: Although they perform well for small-scale image transformations (e.g., changing intensities and colors of pixels), they fail for large-scale ones, such as rotations. This shows the need for methods such as STNs or our framework. In addition, it shows the advantage of explicit transformations as calculated by our framework: Instead of hallucinating corresponding samples of the other domain, possibly worsening classification performance, among others due to blurring, transformation matrices offer invertible mappings, and can be initialized to meaningful values, or the identify function, at least. Performance of CORAL increases with growing b, but still does not reach that of fine-tuning or our models. Interestingly, applying a dataset-global transformation does not increase performance over a local one, although data was generated with such a global transformation. This could show slight advantages of instance-specific transformations, even in this setting, e.g. due to class-specific classification differences and needs.

C. Toy Sequence

As this problem serves as preparation and motivation for the real-world problem of predicting lane changes, we here only consider fine-tuning as baseline.

Upon inspection of Table II we find our earlier assumption confirmed: a model trained on more well-behaved data has problems dealing with noisy trajectories, *Frequency* nearly doubles when switching from domain A to B. $T1$ again denotes our framework, in which $\mathbf{C}$ is pre-trained with the identity matrix. $T2$, however, uses the following pre-training target: During follow periods, the target matrix is $\begin{pmatrix} 0 & 0 \\ 0.5 & 1 \end{pmatrix}$, and during lane changes the identity. Due to this, the Converter is trained to output 0.5 (i.e. projecting the car onto the middle of the road) when its belief is lane following, and otherwise leaves the expected lane change unchanged. This causes a smoothing of trajectories during follow periods, making the trained model less subjective to noise, but still enables the correct detection of lane changes.

Results show the effect of this domain knowledge, while achieving similar *Delay* and *Miss* scores as other methods, $T2$ records by far the lowest *Frequency*, even lower than the original model (more on this in the next section). Depending on b, the ordering of fine-tuning and $T1$ changes, this dataset is too small and simplistic to learn meaningful transformations by itself - this is achieved for the next task. Figure 4 shows the output trajectory of the Converter when applied to a lane change. The smoothing effect in the follow period is nicely visible, and the success of the conversion can be

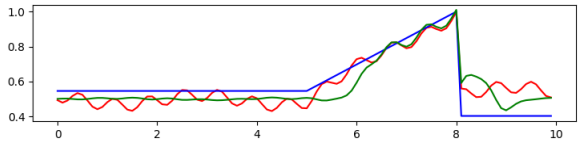

Fig. 4. Visualization of a simulated lane change to the left. The distance to the lane's center line is plotted on the y-axis, time in seconds on the x-axis. The "noisy" lane change from domain B is drawn in red, a corresponding one from domain A in blue (for simplicity, just one of the n is shown). The output of the Converter (with $T2$) is drawn in green, and shows a very plausible converted lane change.

checked visually. The hidden size of $\mathbf{C}$ is 16.

D. Lane Change Prediction

Table III shows results of all analyzed models when considering the 2 features m and v. A hidden size of 32 is used for $\mathbf{C}$. Results for experiments with more features are similar, especially regarding the ranking of algorithms. It can be noted though, that more features initially cause a greater performance drop when changing domains, as the learned knowledge is more specific, but perform better after the application of transfer learning techniques - after retraining the additional information is helpful. Further, fine-tuning performs worse, as the domain gaps get wider, and training the full model on B better, especially reducing *Frequency*, again due to the amount of available information (compare [20] for more results).

TABLE II

RESULTS OF THE TOY SEQUENCE PROBLEM. SMALLER VALUES FOR *Frequency*, *Delay* AND *Miss* ARE BETTER, LARGER ONES FOR *Score*.

b	Model	Frequency	Delay	Miss	Score
100	Fine-tune	3.196	0.945	0.121	0.186
(1)	$T1$	2.849	**0.897**	**0.112**	**0.275**
	$T2$	**2.724**	1.077	0.153	0.262
500	Fine-tune	3.456	0.888	**0.113**	0.137
(5)	$T1$	3.156	**0.878**	0.117	0.203
	$T2$	**1.410**	1.140	0.140	**0.565**
2000	Fine-tune	2.940	**0.903**	0.106	0.257
(20)	$T1$	3.871	0.931	0.120	0.034
	$T2$	**1.300**	0.992	**0.104**	**0.625**
10000	A on B	4.085	0.851	0.108	-
(100)	B on B	2.129	0.542	0.062	-

The pre-training targets in Step 1 are the same as in the previous section (for follow periods the target values of m and v are neutral, i.e. 0.5 and 0). For this task, our proposed models outperform all others, both with $T1$ and $T2$, in terms of *Frequency* and total score. We note, that also $T1$ works well in this context, the Converter is able to gain enough knowledge to find meaningful transformations by itself. Furthermore, also for this task domain knowledge helps improving performance. Our models even perform "better" in terms of *Frequency* than the full model trained on B. This can be explained by the fact that training is done by just using cross-entropy loss, and not by directly optimizing for any higher-level metrics. The found solution thus is a local optimum w.r.t. to this loss, resulting in a well-rounded solution. This shows other exciting properties of our framework: Potentially, it is not only applicable to transfer learning tasks in a strict sense, but also to tune and alter existing models in a desired way. Depending on pre-training, we could nudge models to prioritize different aspects.

When comparing our model to the baselines, we see a bigger difference for pretraining target $T1$. This makes sense, as all models profit from prior information ($T2$) and improve, and with less information given the selected correspondence pairs and the resulting correspondence loss is more valuable.

Thus, in this scenario the correspondence pairs and linked correspondence loss does help improve performance. In particular, our full model outperforms all ablation studies (Mode 0 and Mode 2), and all other baselines.

Figure 5 shows a sample sequence of aligned lane changes, plotting m and v as well as predictions and ground truth labels, comparing our model to fine-tuning only.

CORAL yields somewhat acceptable results, but drops behind fine-tuning. One reason could be the small size of the covariance matrix, and that for these high-level sequential, non-image inputs, domain differences express differently than via pure second-order statistics. Our used pix2pix adaption for this problem consists of simple, one-layered RNNs as generator (hidden size 32) and discriminator (hidden size 8). Arguably, this could be improved, but we found adversarial

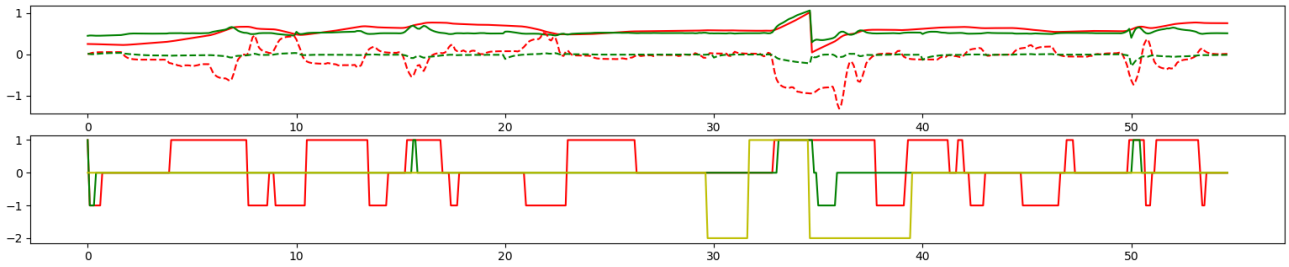

Fig. 5. Visualization of a lane change to the left, on the x-axis time in seconds is plotted. The top plot shows m, once the raw data from domain B (red) and once the Converter's output (green). Same holds for v, this is indicated by the dashed lines. In the bottom plot the labels of the corresponding time steps are shown: The ground truth label is drawn in yellow, the prediction of the fine-tuning approach in red, the output of our model ($T2$) in green. 1 / -1 denote lane changes to the left / right, 0 follow periods and -2 ignore labels, which are inserted between follow and lane change labels and after lane changes, to give the models time to reset. The effects of the Converter can clearly be seen, smoothing out fluctuations and scaling down extreme values of the input features, especially during follow periods. Our model outperforms standard fine-tuning, having much less false predictions while virtually predicting identically during lane changes.

TABLE III

RESULTS OF THE LANE CHANGE PREDICTION PROBLEM. SMALLER VALUES FOR *Frequency*, *Delay* AND *Miss* ARE BETTER, LARGER ONES FOR *Score*.

b	Model	Frequency	Delay	Miss	Score
100	Fine-tune	7.344	0.612	0.008	0.352
	CORAL	7.996	**0.601**	0.011	0.291
	pix2pix	5.399	0.837	0.039	0.439
	Imp	4.363	0.837	0.006	0.592
(1.8)	Mode 0 - $T1$	7.48	0.637	0.008	0.339
	Mode 0 - $T2$	5.477	0.738	0.008	0.502
	Mode 2 - $T1$	5.835	0.699	**0.005**	0.48
	Mode 2 - $T2$	5.243	0.833	0.011	0.51
	Ours - $T1$	4.573	0.797	0.006	0.578
	Ours - $T2$	**3.373**	0.956	**0.005**	**0.672**
500	Fine-tune	7.744	**0.544**	0.01	0.319
	CORAL	8.336	0.547	0.011	0.265
	pix2pix	6.003	0.842	0.071	0.321
	Imp	4.476	0.835	0.010	0.576
(9.1)	Mode 0 - T1	7.93	0.559	0.01	0.302
	Mode 0 - $T2$	5.138	0.704	0.006	0.536
	Mode 2 - $T1$	5.635	0.668	0.003	0.502
	Mode 2 - $T2$	5.535	0.658	0.003	0.511
	Ours - $T1$	4.848	0.759	**0.002**	0.565
	Ours - $T2$	**3.524**	0.887	**0.002**	**0.669**
1000	Fine-tune	6.551	0.620	0.005	0.424
	CORAL	6.933	**0.608**	0.010	0.383
	pix2pix	2.339	1.157	0.144	0.460
	Imp	3.803	0.800	0.006	0.641
(18.2)	Mode 0 - $T1$	6.629	0.617	0.005	0.418
	Mode 0 - $T2$	5.223	0.706	0.005	0.531
	Mode 2 - $T1$	5.058	0.686	**0.002**	0.552
	Mode 2 - $T2$	4.976	0.702	0.003	0.556
	Ours - $T1$	3.594	0.901	0.01	0.646
	Ours - $T2$	**3.241**	0.888	0.003	**0.691**
5500 (100)	A on B	11.672	0.347	0.010	-
	B on B	4.732	0.698	0.005	-

training on trajectories to be tedious and non-promising. Our implicit variant fares better, matching our full model with $T1$. This shows, on the one hand, that such a domain adaptation can be done implicitly, as well. On the other hand this proves, that we do not loose expressiveness by using transformation matrices. We would like to remind the reader of the advantages of such a scheme as described in the introduction, such as better verifiability, and the possibility of initializing $\mathbf{T}$ with domain knowledge: No method can compare with $T2$, especially for smaller b.

VII. CONCLUSION

We have proposed a general framework for transfer learning, applying it to image and sequential data, and performing better than existing methods. Primarily, we analyze the problem of predicting lane changes, for which we switch domains from proprietary fleet data to a public dataset. While doing so, our method shows a measurable smoothing of otherwise too sensitive prediction results, significantly reducing especially the false positive rate. Using an explicit transformation matrix comes with benefits for many applications, such as the explainability of the method and the possibility to integrate often existing domain knowledge. We adapt the pix2pix model to sequences by using simple LSTMs of hidden size 32 and 8 for generator and discriminator, respectively. Although this could surely be improved, we found adversarial generation of trajectories tedious and leave this for future work.

REFERENCES

[1] J. Zhu, T. Park, P. Isola, and A. A. Efros, "Unpaired image-to-image translation using cycle-consistent adversarial networks," in *Int. Conf. on Computer Vision (ICCV)*, 2017.

[2] W. Aswolinskiy and B. Hammer, "Unsupervised transfer learning for time series via self-predictive modelling-first results," in *Workshop on New Challenges in Neural Computation (NC2)*, 2017.

[3] B. Paaßen, A. Schulz, and B. Hammer, "Linear supervised transfer learning for generalized matrix lvq," in *Workshop on New Challenges in Neural Computation (NC2)*, 2016.

[4] H. Noh, P. H. Seo, and B. Han, "Image question answering using convolutional neural network with dynamic parameter prediction," in *Int. Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[5] S. Antol, A. Agrawal, J. Lu, M. Mitchell, D. Batra, C. Lawrence Zitnick, and D. Parikh, "Vqa: Visual question answering," in *Int. Conf. on Computer Vision (ICCV)*, 2015.

[6] G. Hinton, O. Vinyals, and J. Dean, "Distilling the knowledge in a neural network," *Advances in Neural Information Processing Systems (NIPS)*, 2014.

[7] J. Yim, D. Joo, J. Bae, and J. Kim, "A gift from knowledge distillation: Fast optimization, network minimization and transfer learning," in *Int. Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[8] K. Bousmalis, N. Silberman, D. Dohan, D. Erhan, and D. Krishnan, "Unsupervised pixel-level domain adaptation with generative adversarial networks," in *Int. Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[9] X. Ouyang, Y. Cheng, Y. Jiang, C.-L. Li, and P. Zhou, "Pedestrian-synthesis-gan: Generating pedestrian data in real scene and beyond," *arXiv preprint arXiv:1804.02047*, 2018.

[10] P. Isola, J. Zhu, T. Zhou, and A. A. Efros, "Image-to-image translation with conditional adversarial networks," *Int. Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[11] X. Li, S. Liu, J. Kautz, and M.-H. Yang, "Learning linear transformations for fast arbitrary style transfer," *arXiv preprint arXiv:1808.04537*, 2018.

[12] M. Jaderberg, K. Simonyan, A. Zisserman, *et al.*, "Spatial transformer networks," in *Advances in neural information processing systems (NIPS)*, 2015, pp. 2017–2025.

[13] L. Duan, D. Xu, and I. W. Tsang, "Learning with augmented features for heterogeneous domain adaptation," in *Int. Conf. on Machine Learning (ICML)*, 2012.

[14] B. Paaßen, A. med. Schulz, and B. Hammer, "Linear supervised transfer learning for generalized matrix lvq," in *Workshop New Challenges in Neural Computation*, 2016.

[15] B. Sun, J. Feng, and K. Saenko, "Return of frustratingly easy domain adaptation," in *AAAI Conference on Artificial Intelligence*, 2016.

[16] F. Schroff, D. Kalenichenko, and J. Philbin, "Facenet: A unified embedding for face recognition and clustering," in *Int. Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2015.

[17] J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation," *Int. Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2015.

[18] F. A. Gers, J. Schmidhuber, and F. Cummins, "Learning to forget: Continual prediction with lstm," *Neural Computation*, 1999.

[19] Y. LeCun and C. Cortes, "MNIST handwritten digit database," 2010.

[20] O. Scheel, N. Shankar Nagaraja, L. Schwarz, N. Navab, and F. Tombari, "Attention-based lane change prediction," *Int. Conf. on Robotics and Automation (ICRA)*, 2019.

[21] J. Schlechtriemen, A. Wedel, J. Hillenbrand, G. Breuel, and K. Kuhnert, "A lane change detection approach using feature ranking with maximized predictive power," in *Intelligent Vehicles Symposium (IV)*, 2014.

[22] A. Jain, A. Singh, H. S. Koppula, S. Soh, and A. Saxena, "Brain4cars: Car that knows before you do via sensory-fusion deep learning architecture," *Int. Conf. on Robotics and Automation (ICRA)*, 2016.

[23] "Ngsim project," https://ops.fhwa.dot.gov/trafficanalysistools/ngsim.htm.

[24] J. Johnson, A. Alahi, and F. F. Li, "Perceptual losses for real-time style transfer and super-resolution," *arXiv preprint arXiv:1603.08155*, 2016.

[25] M. Mathieu, C. Couprie, and Y. LeCun, "Deep multi-scale video prediction beyond mean square error," *Int. Conf. on Learning Representations (ICLR)*, 2015.