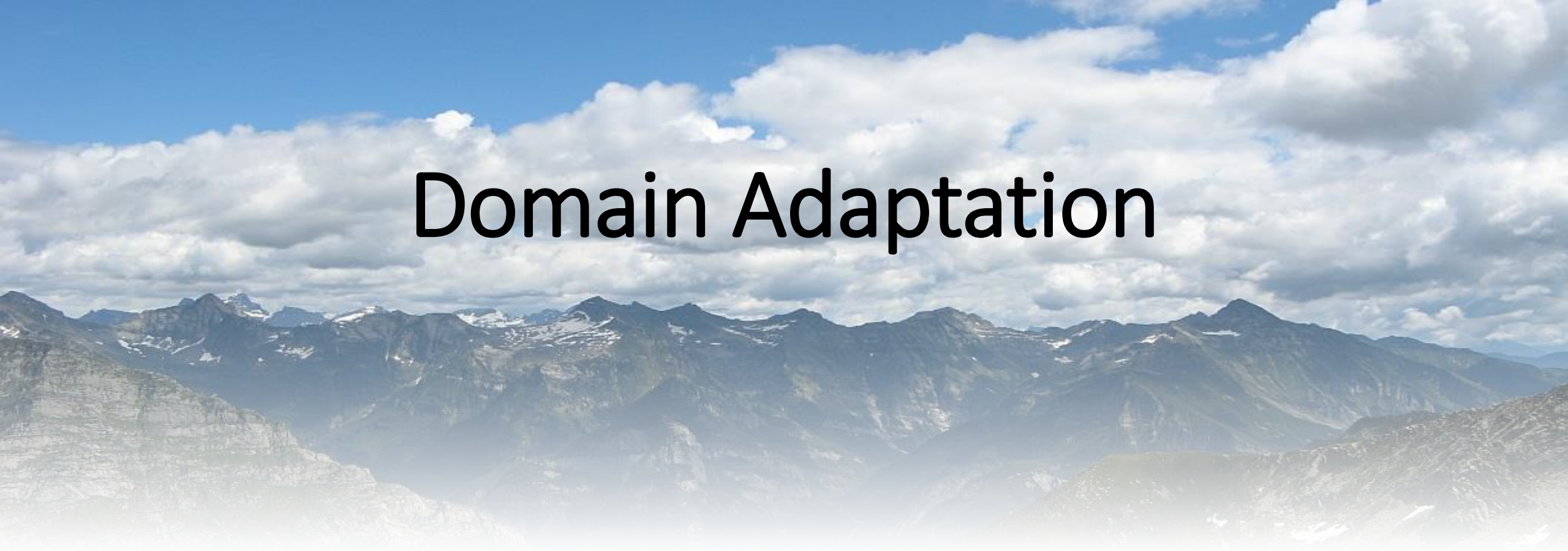


Domain Adaptation

Alessandro Giusti

Dalle Molle Institute for Artificial Intelligence
Lugano, Switzerland

Contact: alessandrog@idsia.ch

https://idsia-robotics.github.io/

IDSIA

1. Motivations

Why do we even care?

1.1 Motivating story 1

An application to robot perception

“*It will be easy*”, they said.

Our focus

Perception of the trail direction

Mapping

Obstacle perception and avoidance

Path Planning

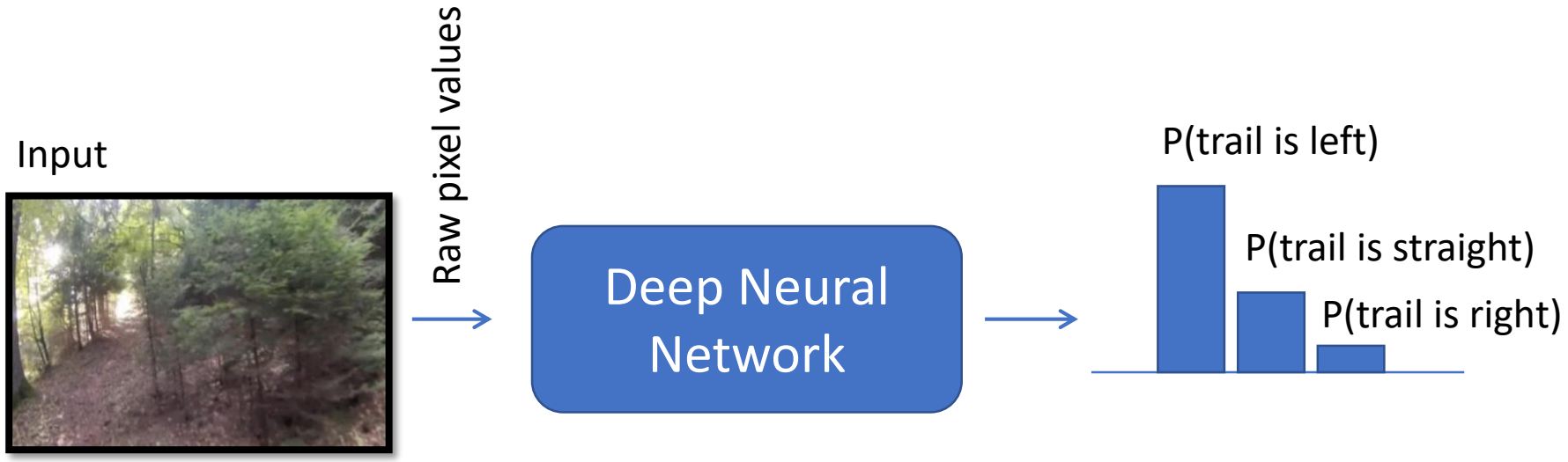
Visual Odometry

Control

A challenging pattern recognition problem

Trail heading left	Trail heading straight ahead	Trail heading right

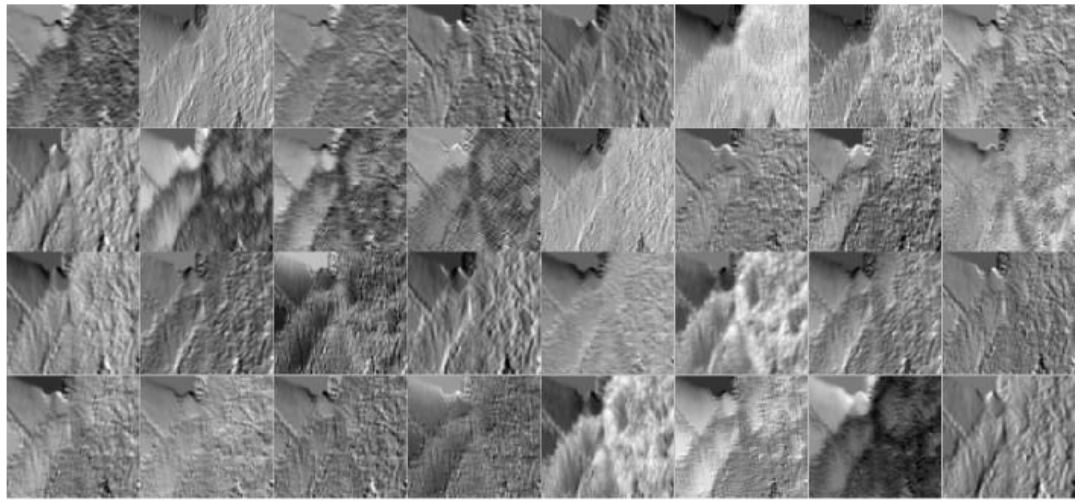
An image classification problem



Deep Network Architecture

L0 - Input layer: 3 maps of 101x101

L1 - Convolutional Layer: 32 maps of 98x98 neurons. Filter: 4x4



L2 - MaxPooling Layer: 32 maps of 49x49 neurons. Kernel 2x2

L3 - Convolutional Layer: 32 maps of 46x46. Filter 4x4

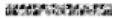
L4 - MaxPooling Layer: 32 maps of 23x23. Kernel: 2x2

L5 - Convolutional Layer: 32 maps of 20x20. Filter: 4x4

L6 - MaxPooling Layer: 32 maps of 10x10 neurons. Kernel: 2x2

L7 - Convolutional Layer: 32 maps of 8x8 neurons. Filter: 4x4

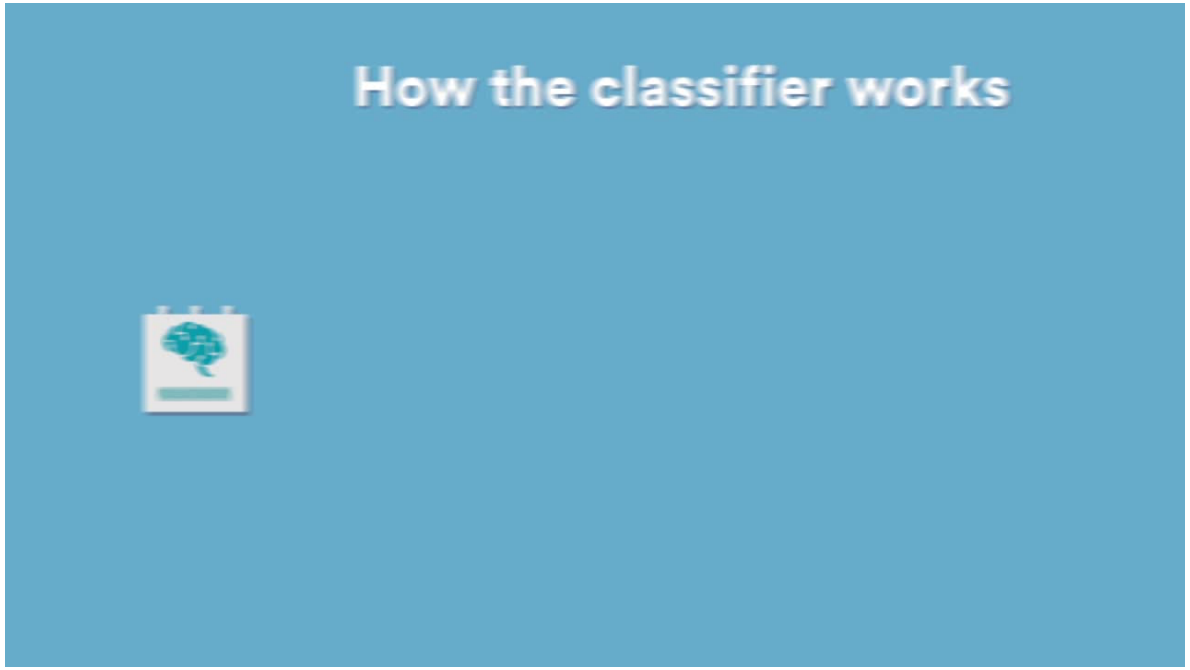
L8 - MaxPooling Layer: 32 maps of 4x4 neurons. Kernel: 2x2



L9 - Fully Connected Layer: 200 neurons

L10 - Output Layer: 3 neurons

Internal view of the classifier



How the classifier was trained

Da

Training the classifier

20^k of images

Classification accuracy

		Predicted Labels		
		TR	GS	TL
Actual Labels	TR	82.05%	1.18%	16.77%
	GS	5.15%	89.78%	5.07%
	TL	13.16%	2.97%	83.87%

(a) DNN+mir (acc: 85.23%)

		Predicted Labels		
		TR	GS	TL
Actual Labels	TR	76.39%	01.13%	22.48%
	GS	04.38%	89.52%	06.10%
	TL	10.79%	02.94%	86.27%

(b) DNN (acc: 84.57%)

Test images which are easily classified

(a) GS frames with highest $P(\text{GS})$, i.e. frames where the path is easily found as being straight ahead

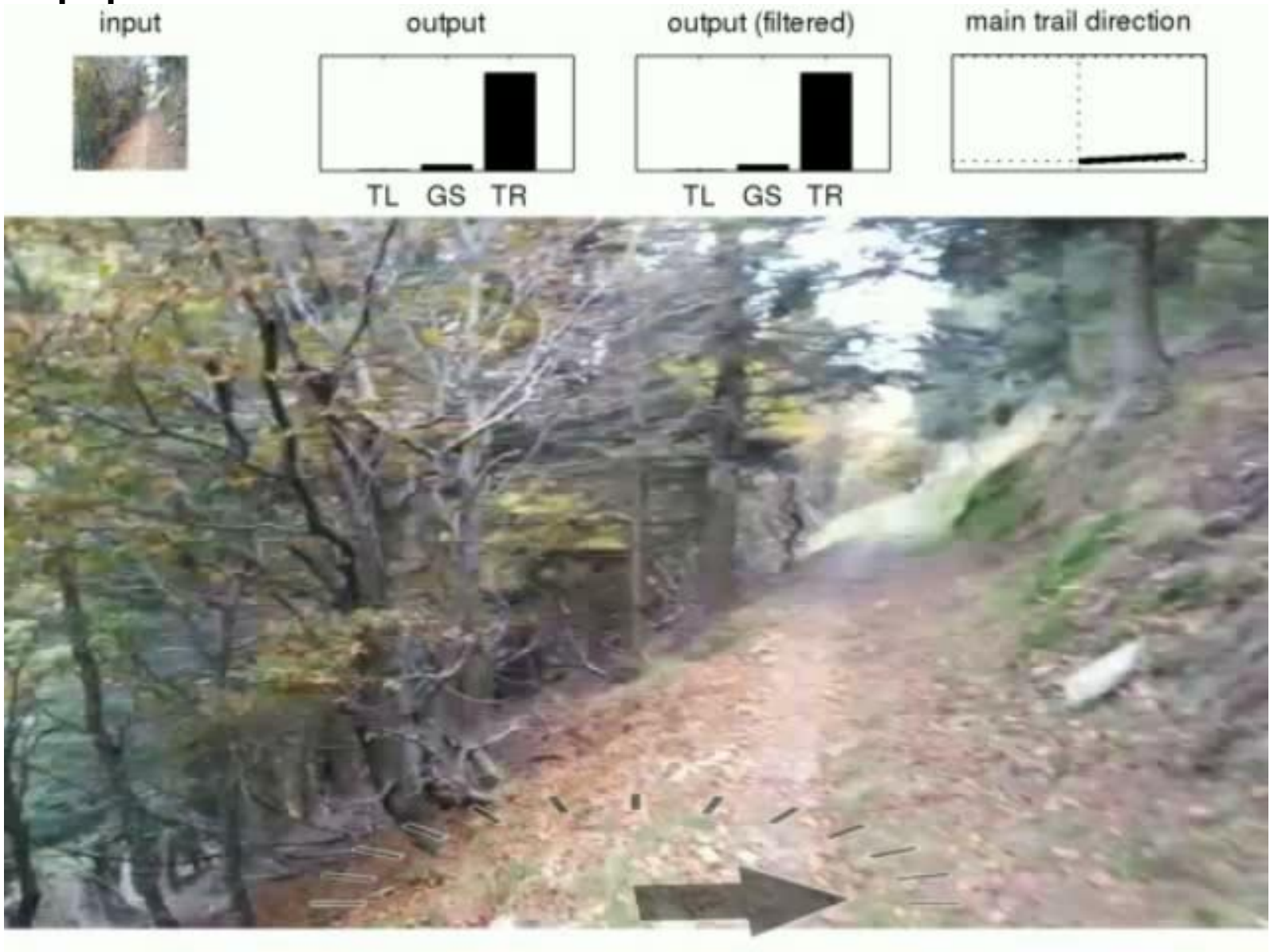
(b) not-GS images with lowest $P(\text{GS})$, i.e. frames where the path is easily found as *not* being straight ahead

Test images which are misclassified

(c) GS images with lowest $P(\mathrm{GS})$, i.e. failure cases where the path should be straight ahead but was not detected as such

(d) not-GS images with highest $P(\mathrm{GS})$, i.e. failure cases where the path is not straight ahead but was detected as such

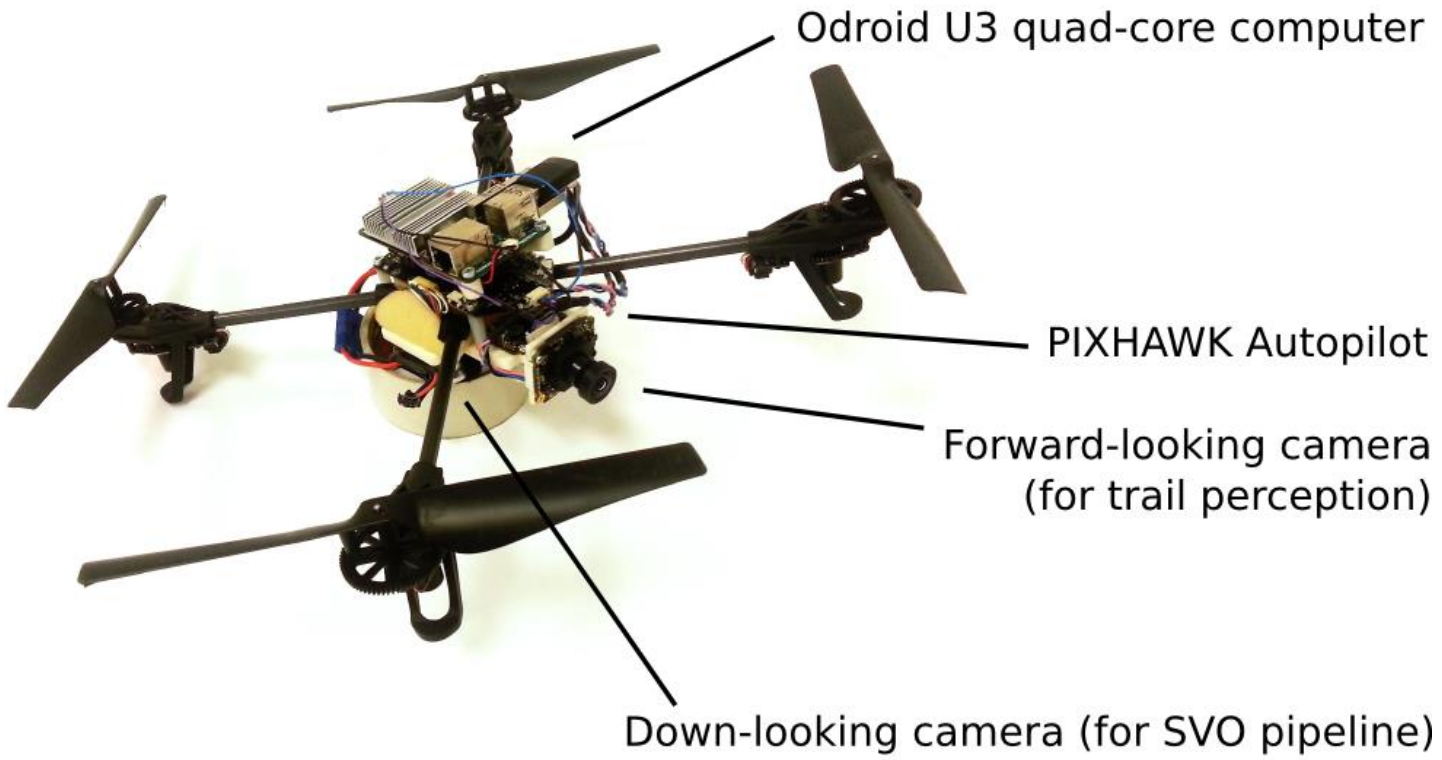
Application to video acquired from an handh



Steering a MAV with a purely reactive controller

Implementation

Odroid U3 runs Semi-direct Visual Odometry (SVO) pipeline and our DNN at more than 15FPS




Where is the problem?

Brainstorm here!

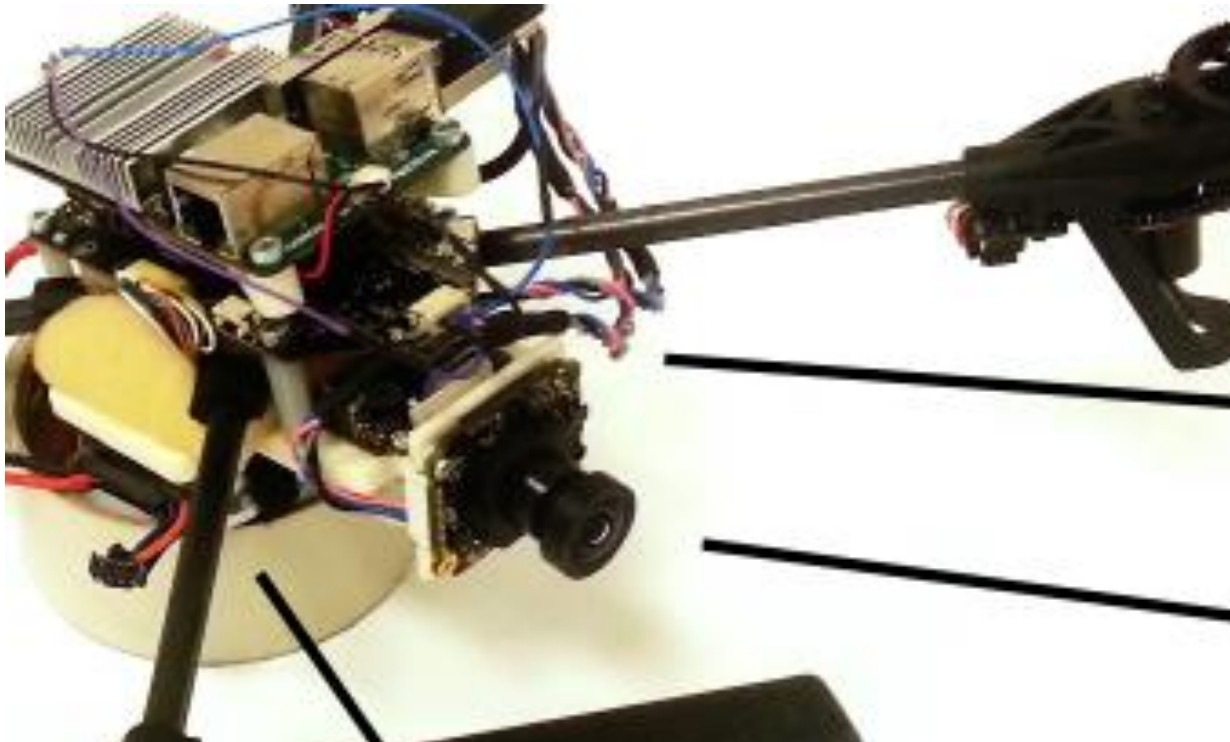
https://answergarden.ch/1279254

Where is the problem?

The devil is in the details

Source domain (a lot of training data)

Target domain (few or no training data)

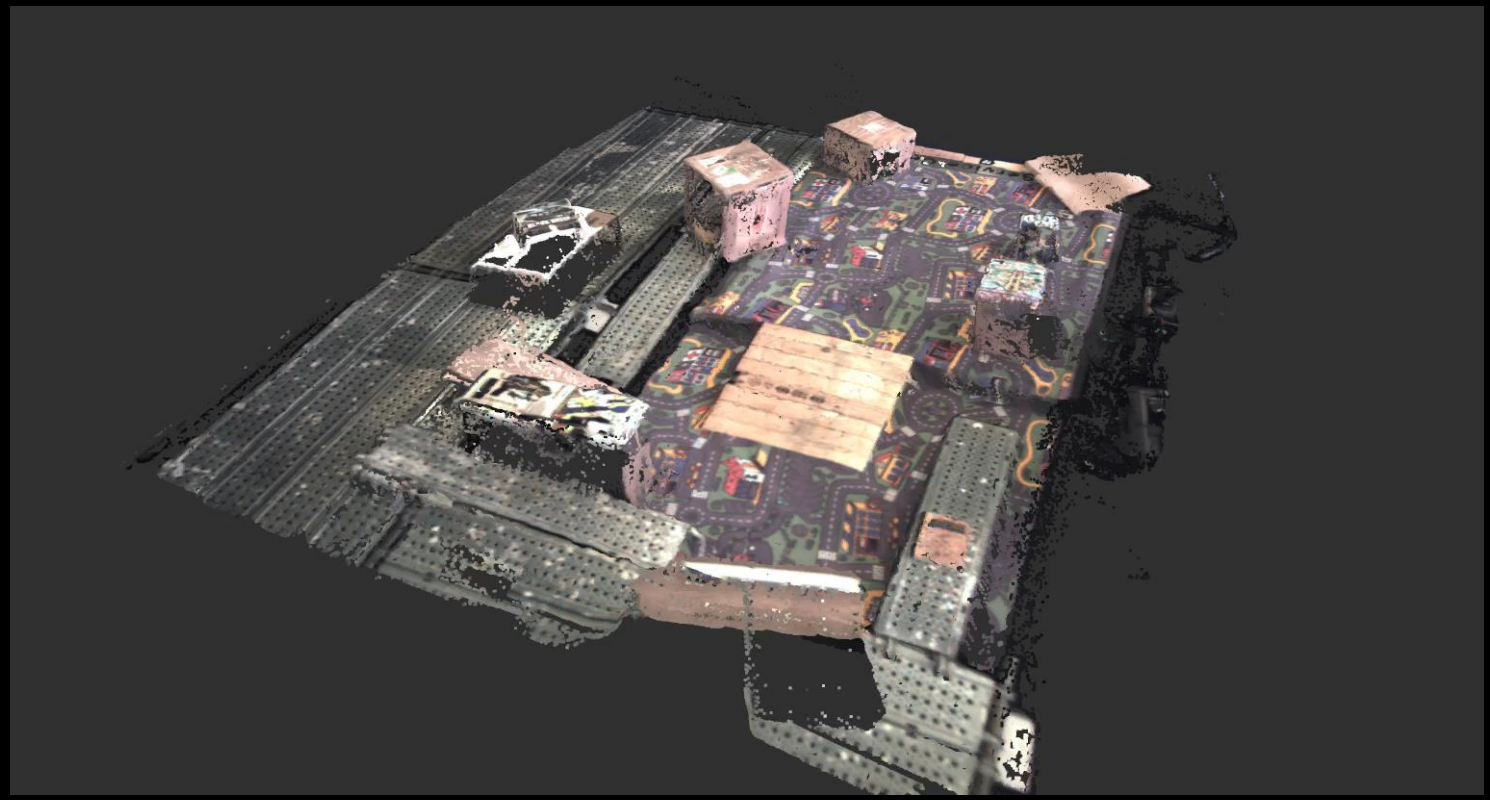


1.2 Motivating story 2

Traversability estimation for ground robots

Traversability rules are non-trivial and robot-dependent

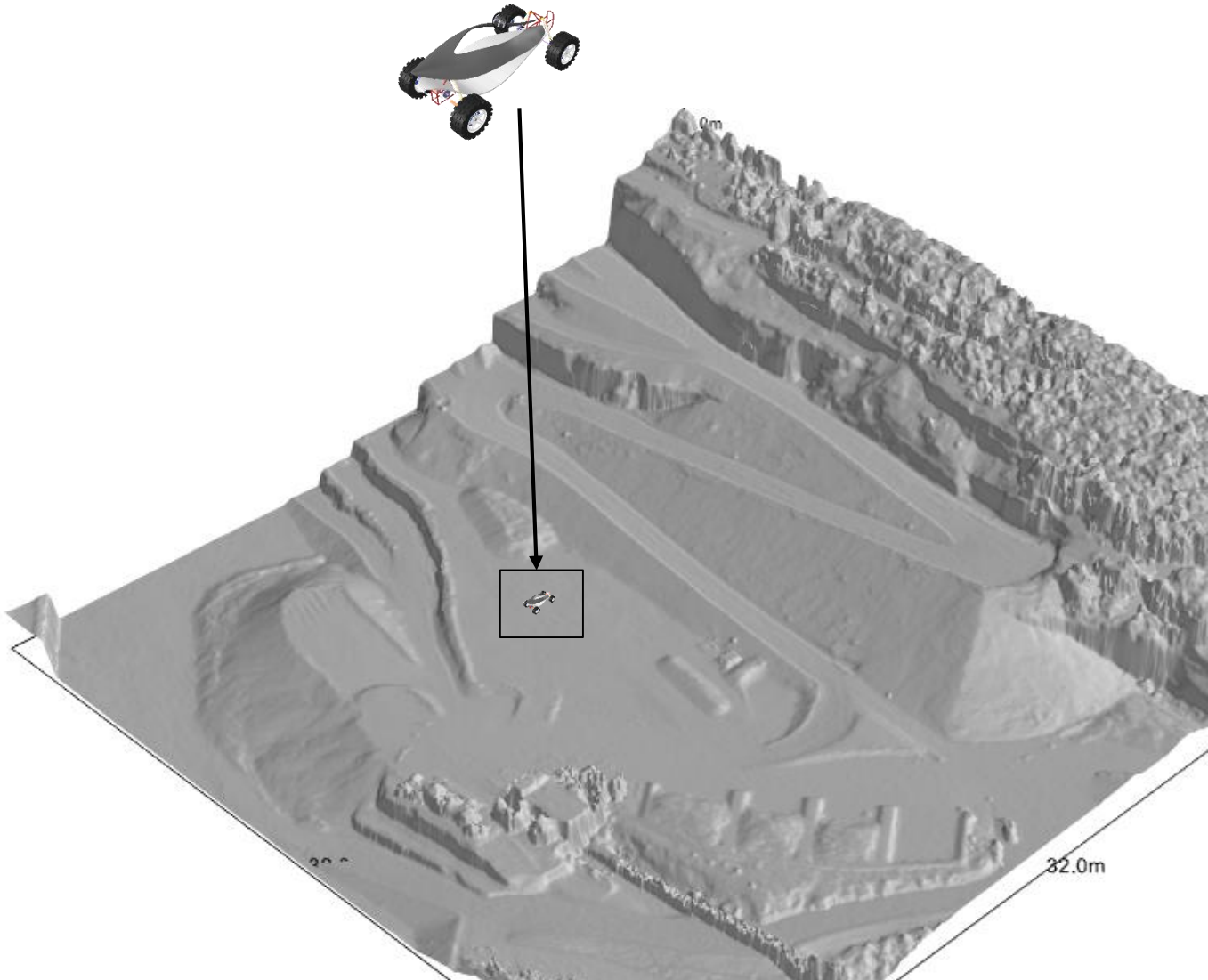
UAVs can create detailed 3D terrain maps



3D reconstruction of scenario at ETH-RSL from UZH-RPG group

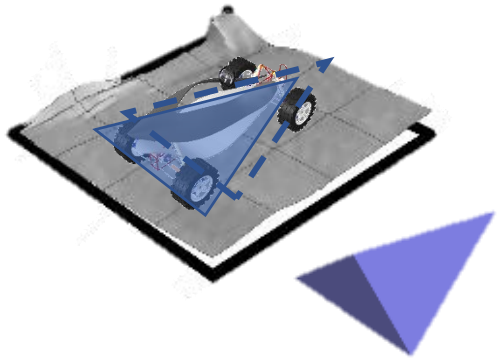
Problem statement

- Given a robot model
- Given a large 3D map
- Where can my robot pass?

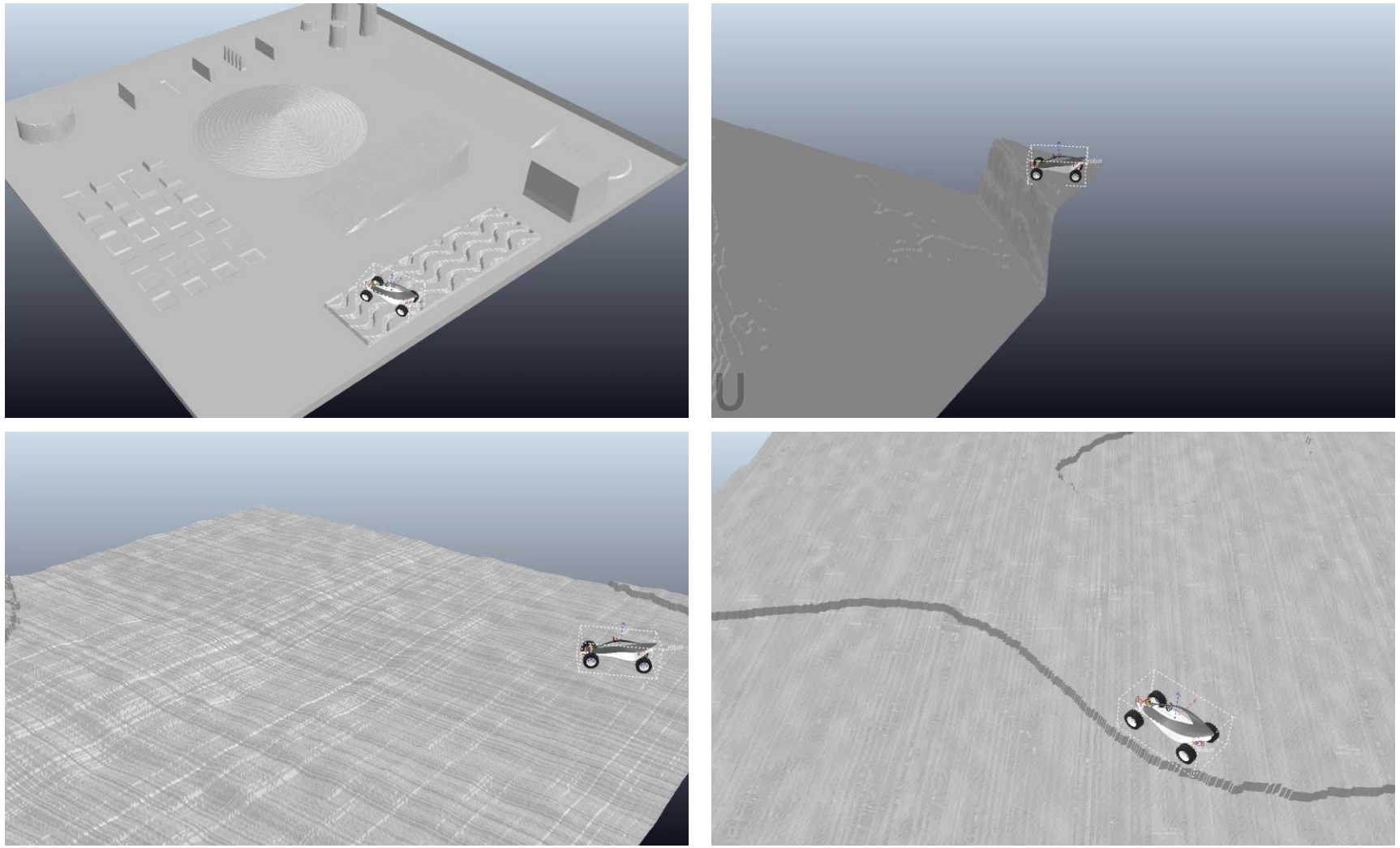


Solution

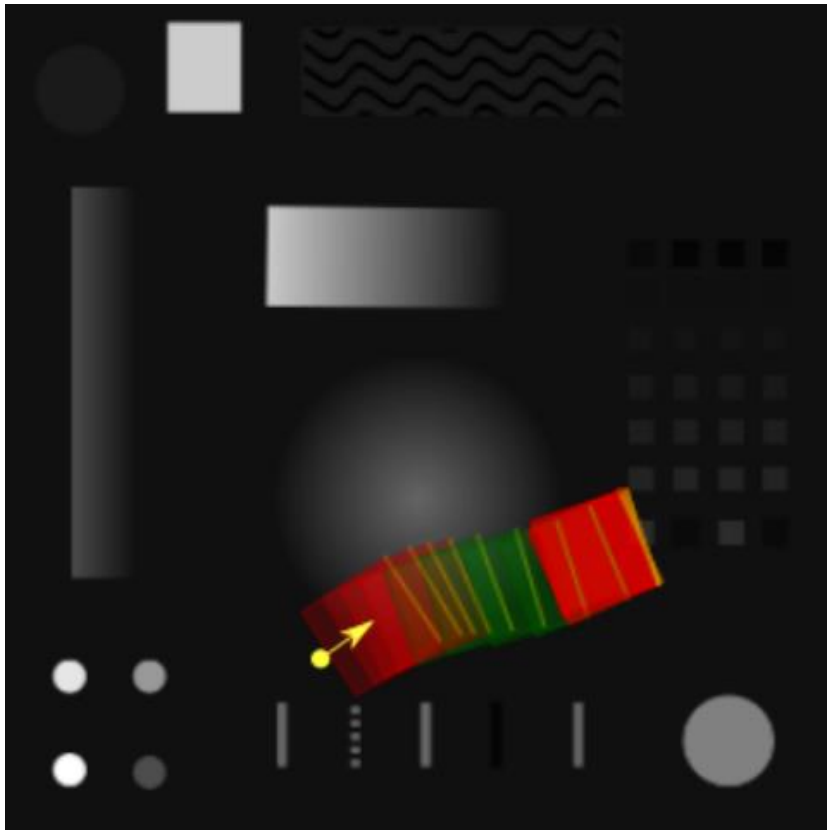
Train a deep neural network that given a small oriented patch of terrain, predicts whether our robot can move from the center to the end of the patch



Gathering training data in simulation

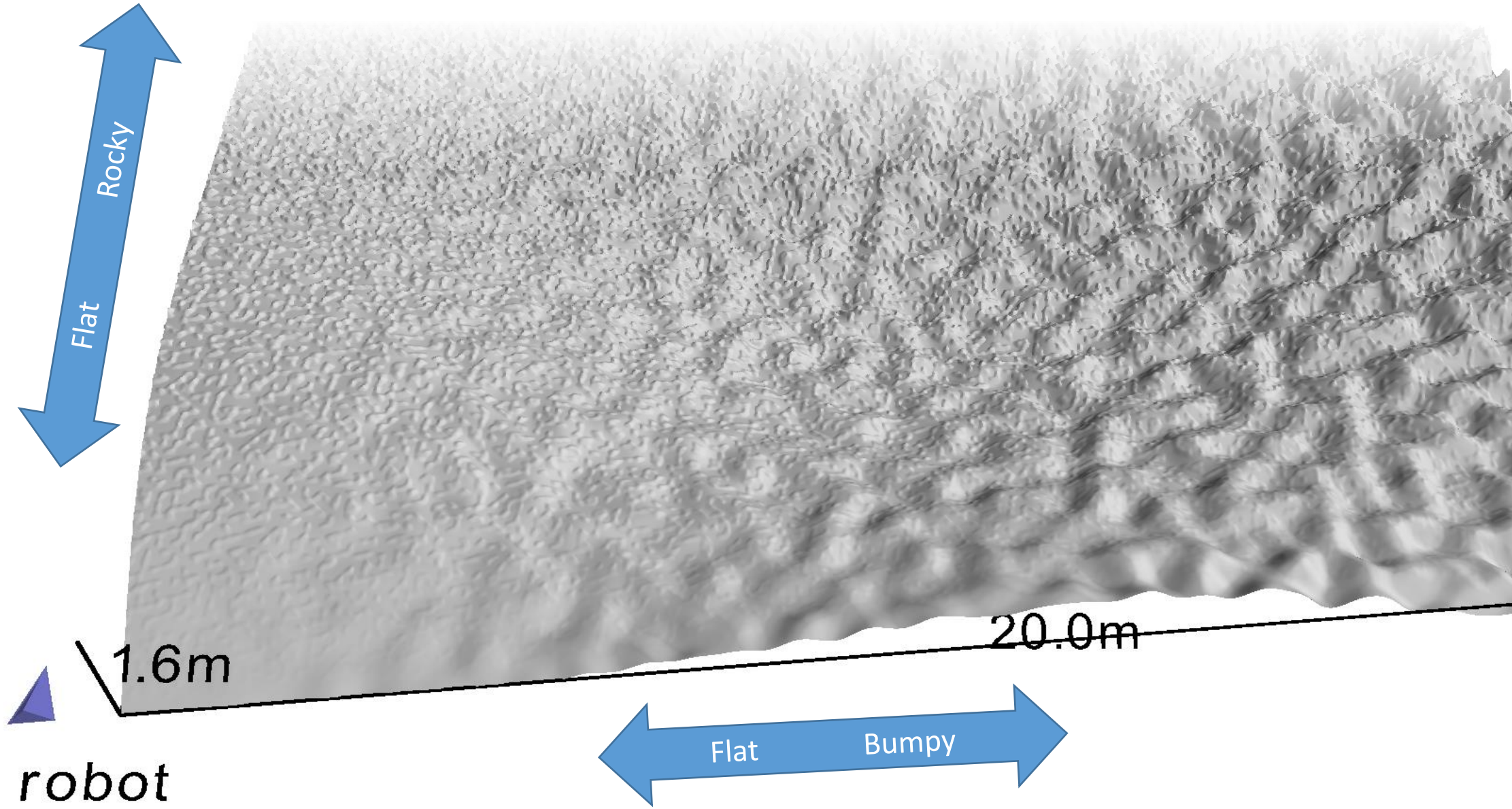


Gathering training data from simulations

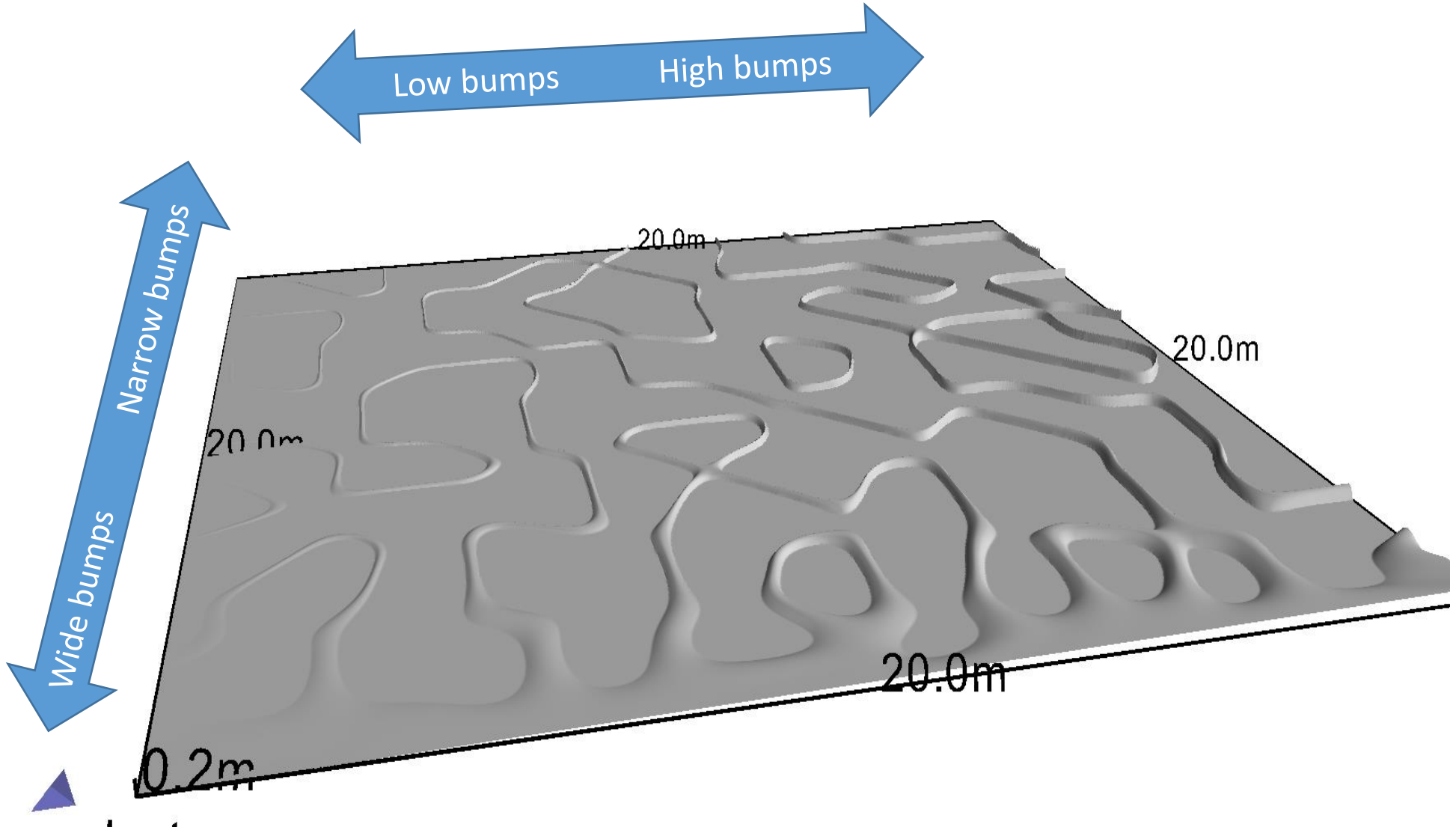


- Spawn robot with random position and orientation
- Let it proceed straight until stuck
- Save traversed / nontraversed patches
- Repeat

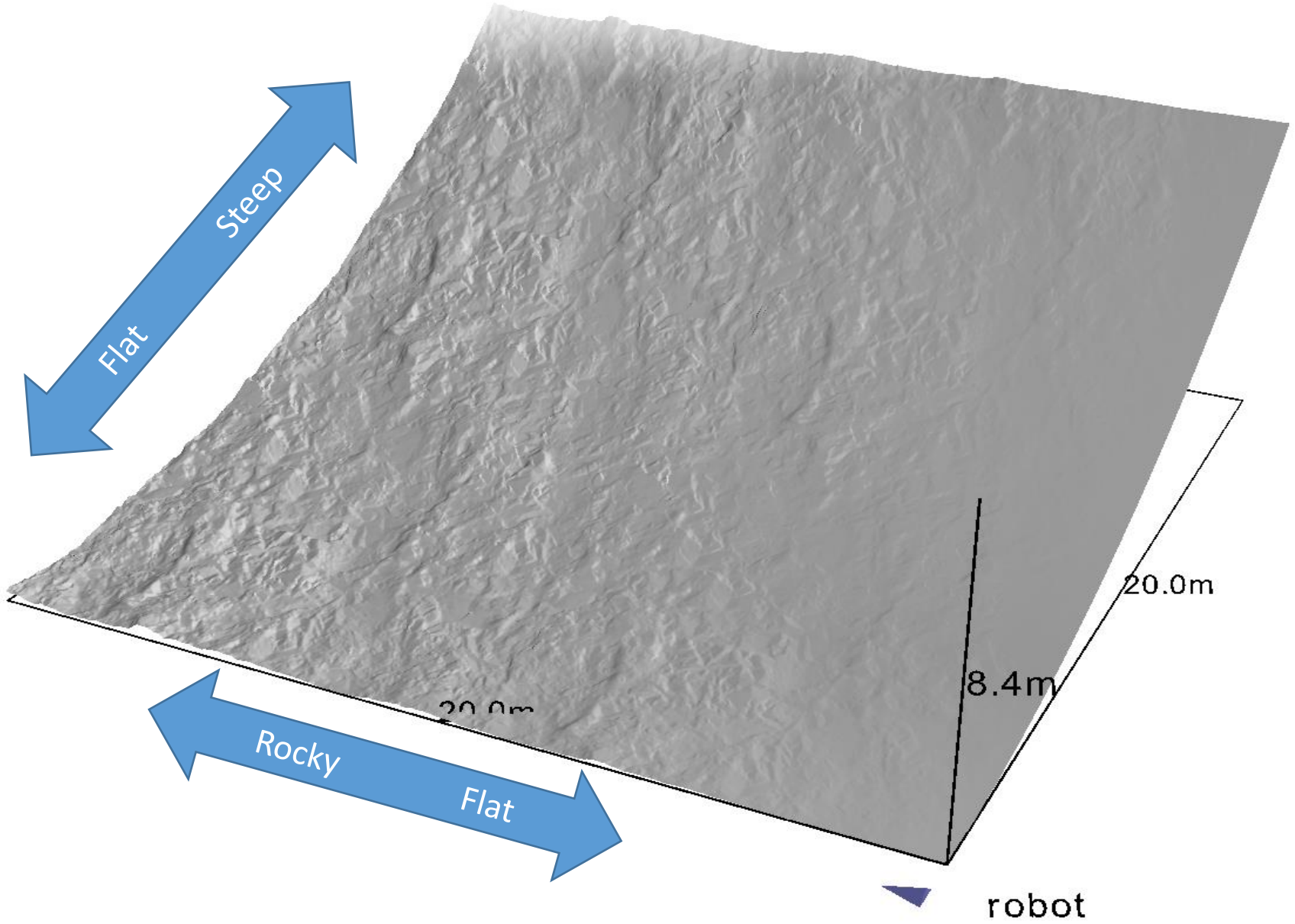
Procedurally-generated training terrains



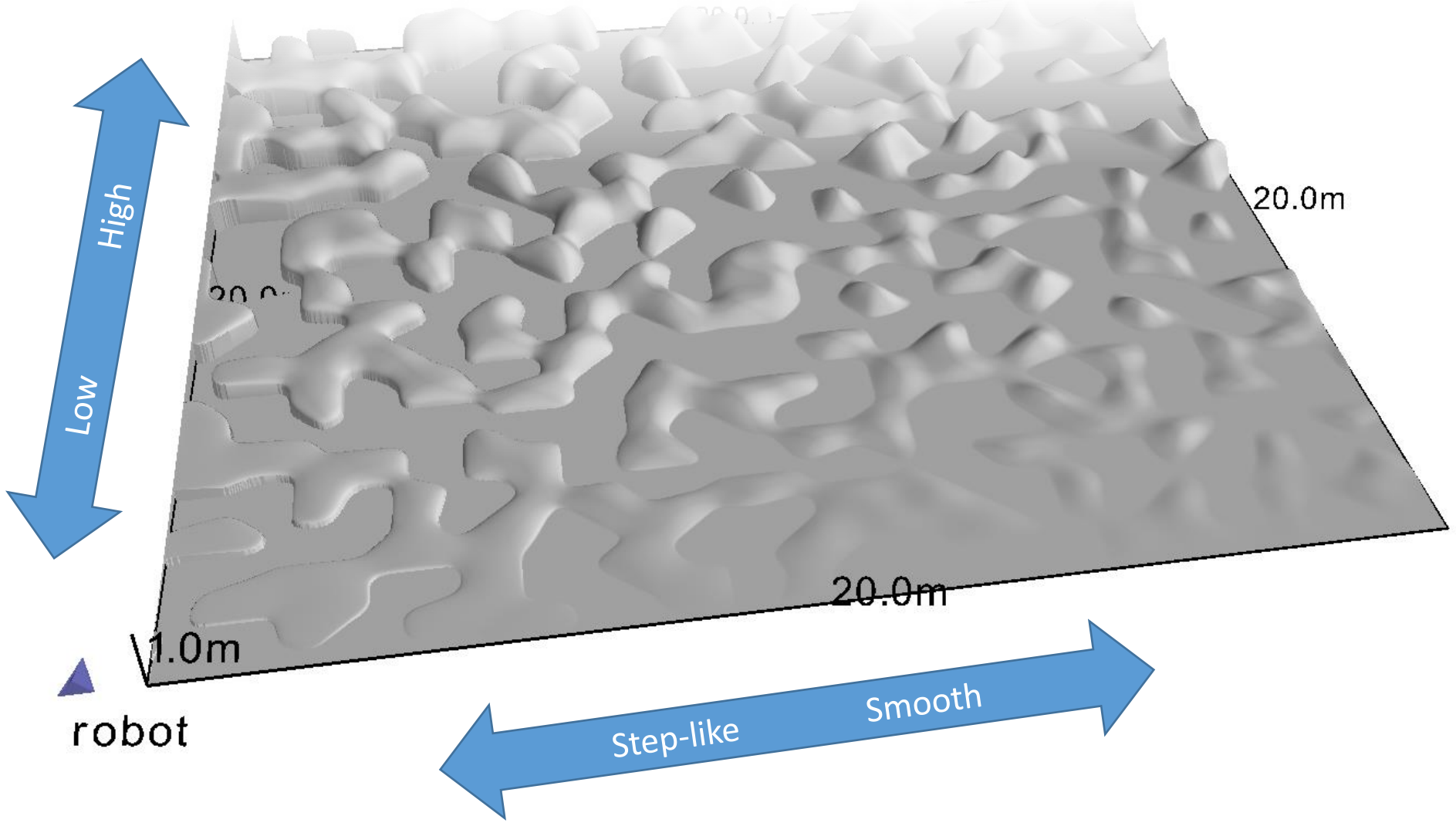
Procedurally-generated training terrains



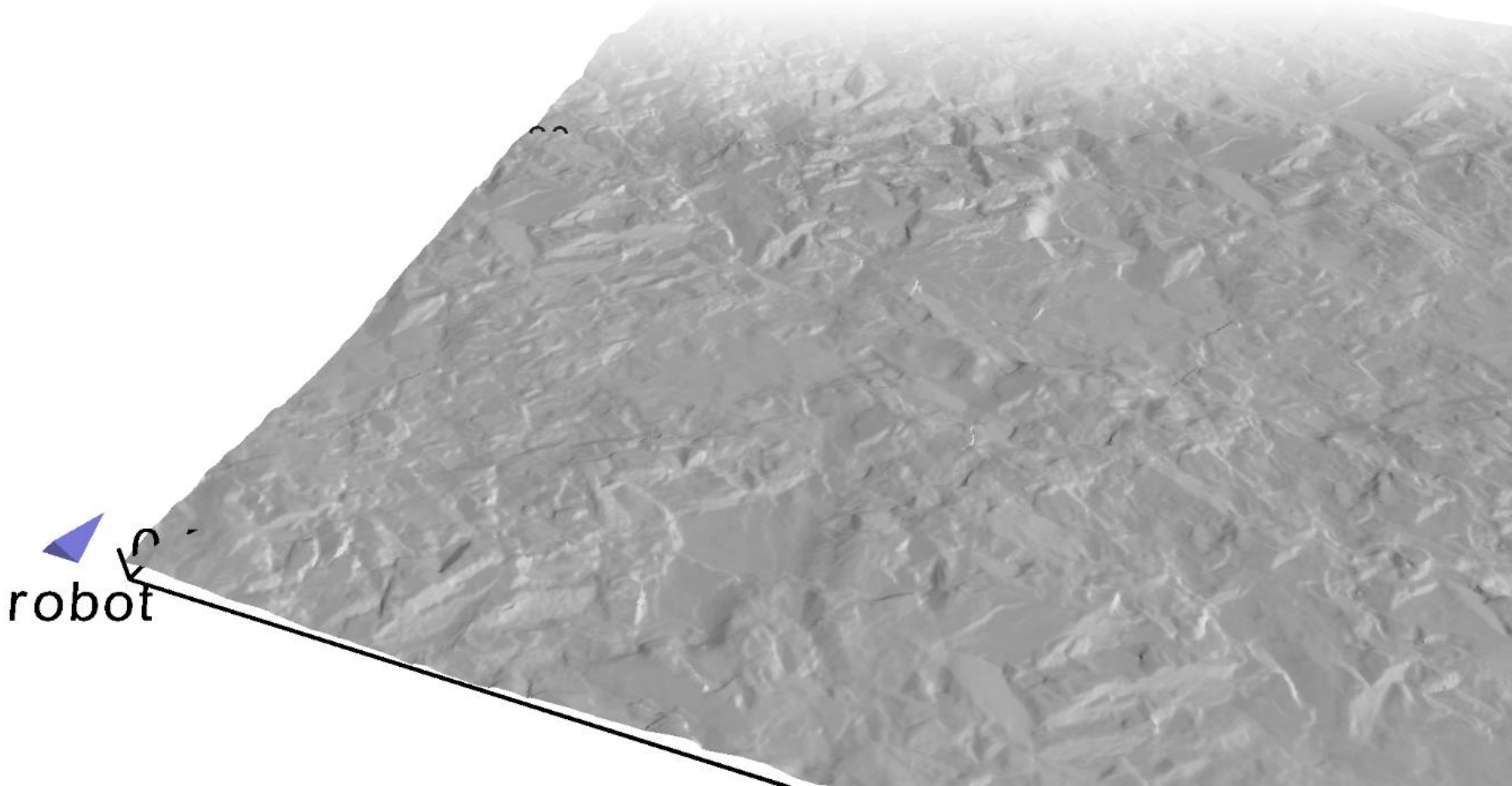
Procedurally-generated training terrains



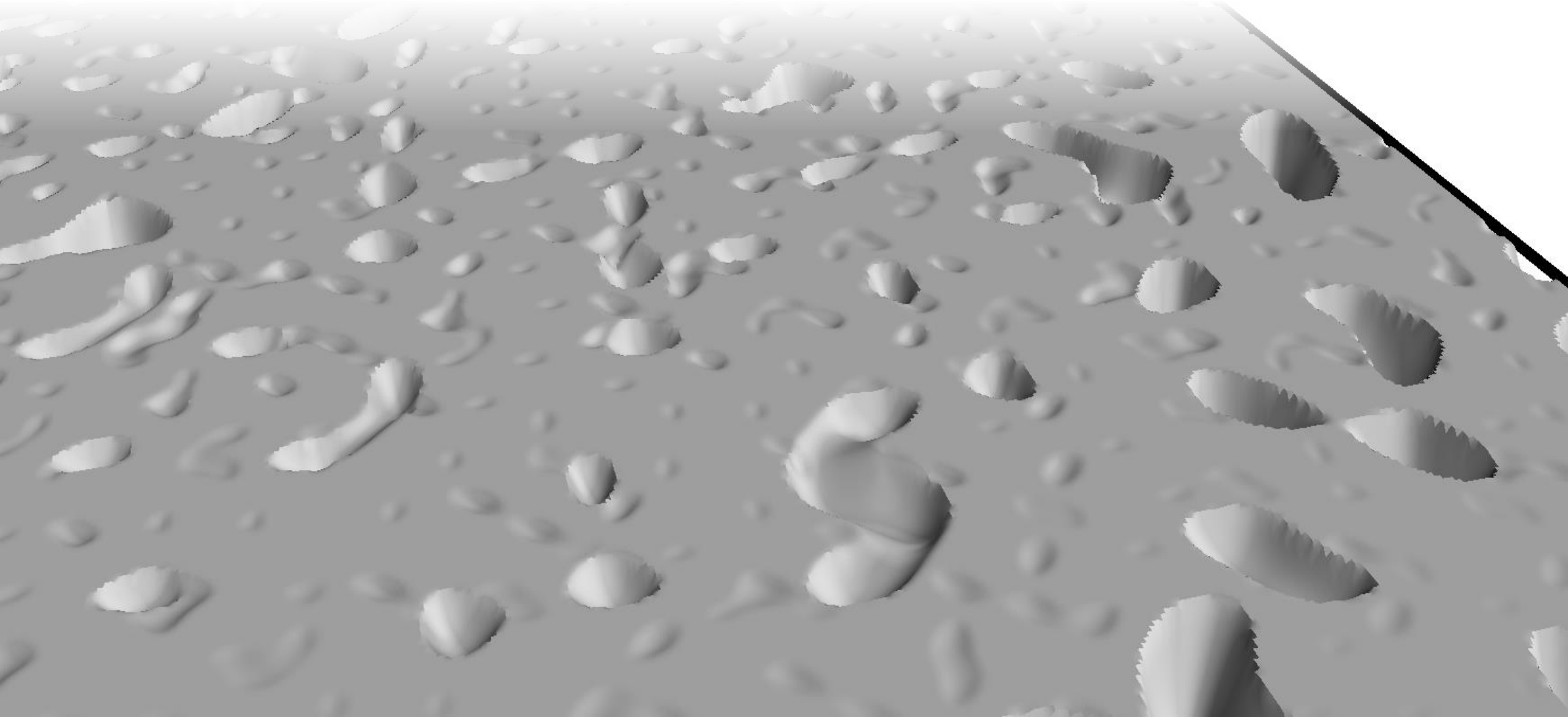
Procedurally-generated training terrains



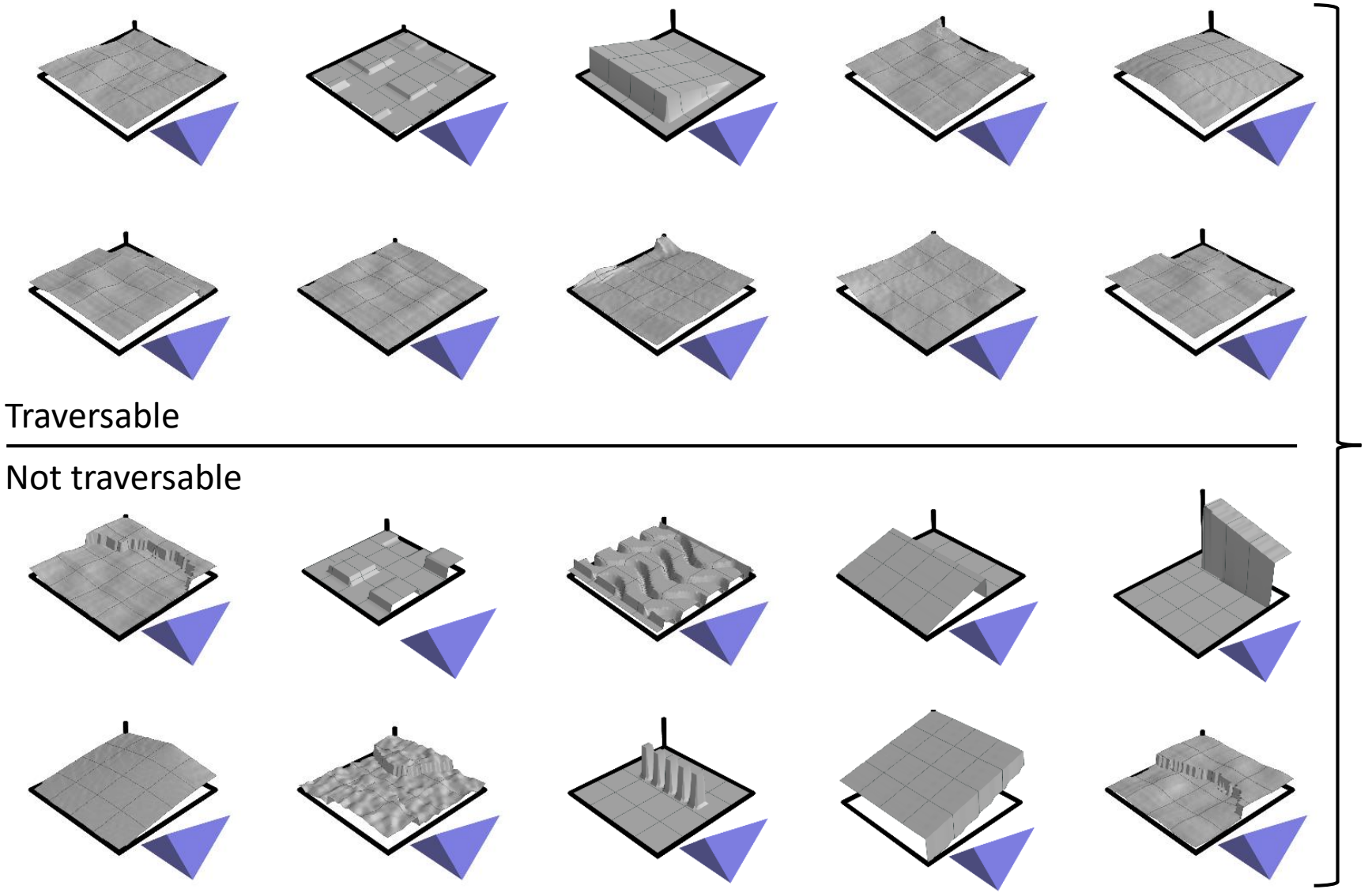
Procedurally-generated training terrains



Procedurally-generated training terrains



Training patches (20 out of 100k)



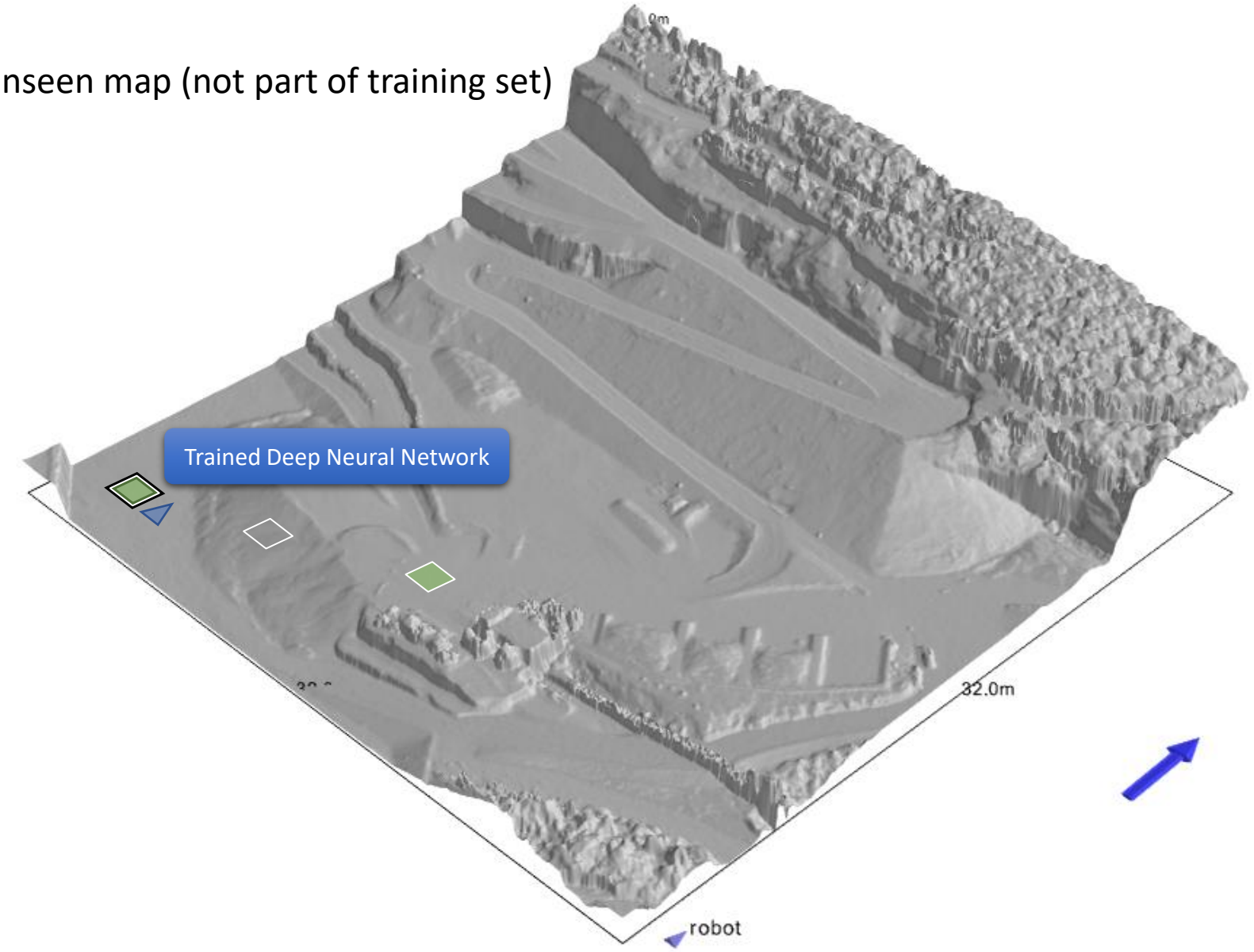
Quantitative results on unseen testing data

	Synthetic dataset		Real-elevation dataset	
	ACC	AUC	ACC	AUC
CNN	0.9134	0.9756	0.7456	0.8729
RandomForest	0.7884	0.9128	0.5940	0.6556
Baseline	0.4956	0.4957	0.5072	0.4974

Chavez et al., *Robotics and Automation Letters, 2018*

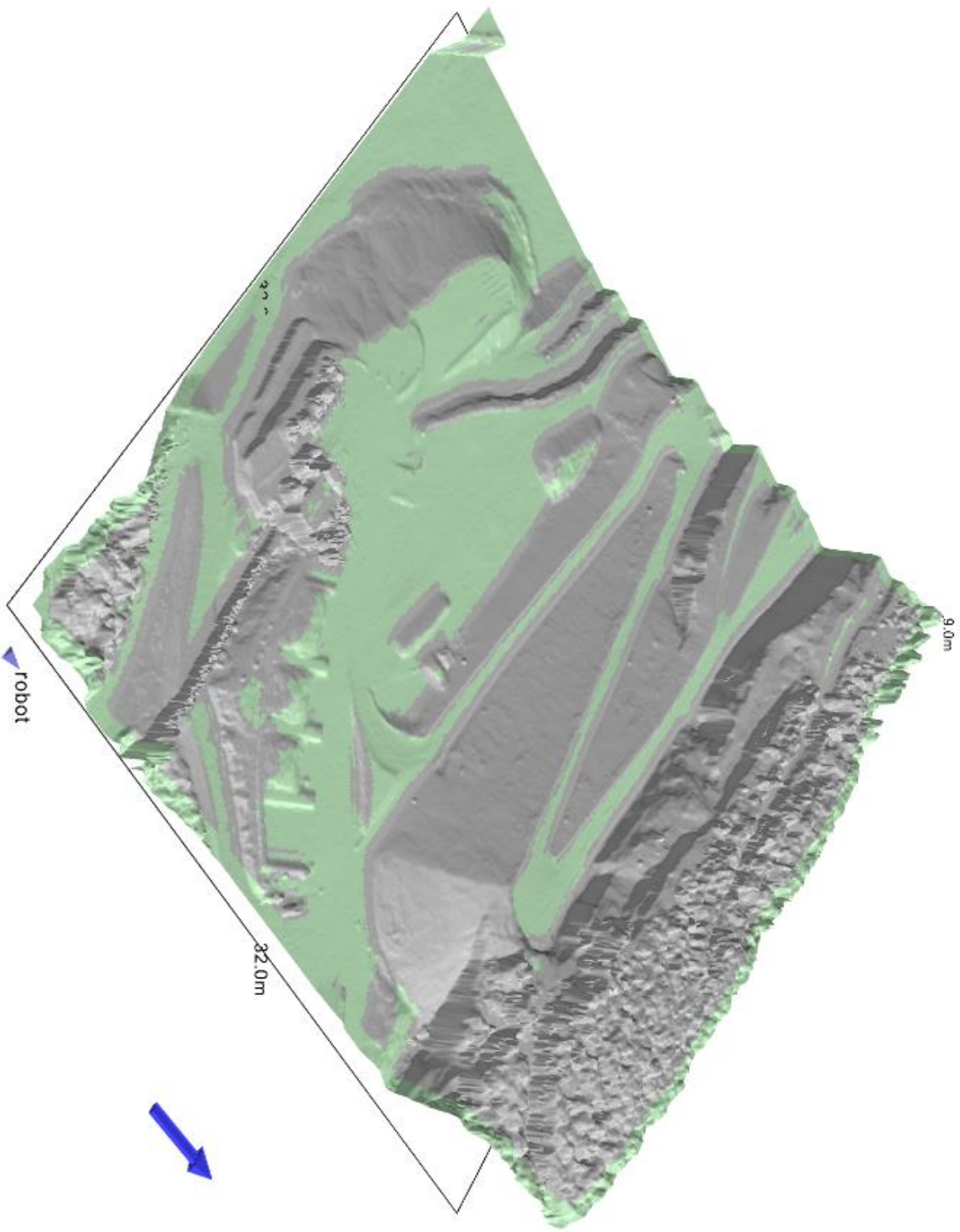
Computing a traversability map

Unseen map (not part of training set)

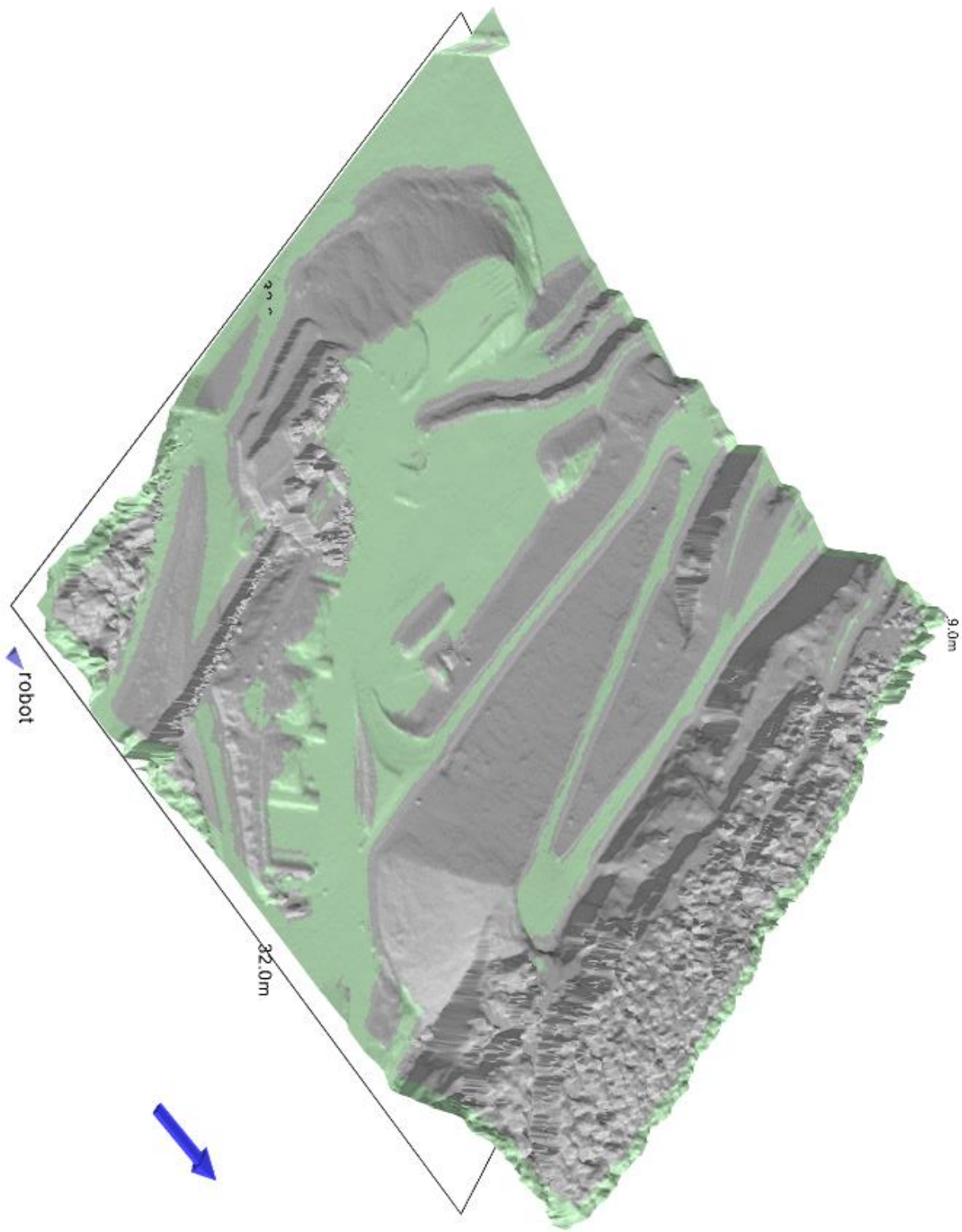


High-resolution DEM data from SenseFly (3D reconstruction of a Swiss Mining Quarry)

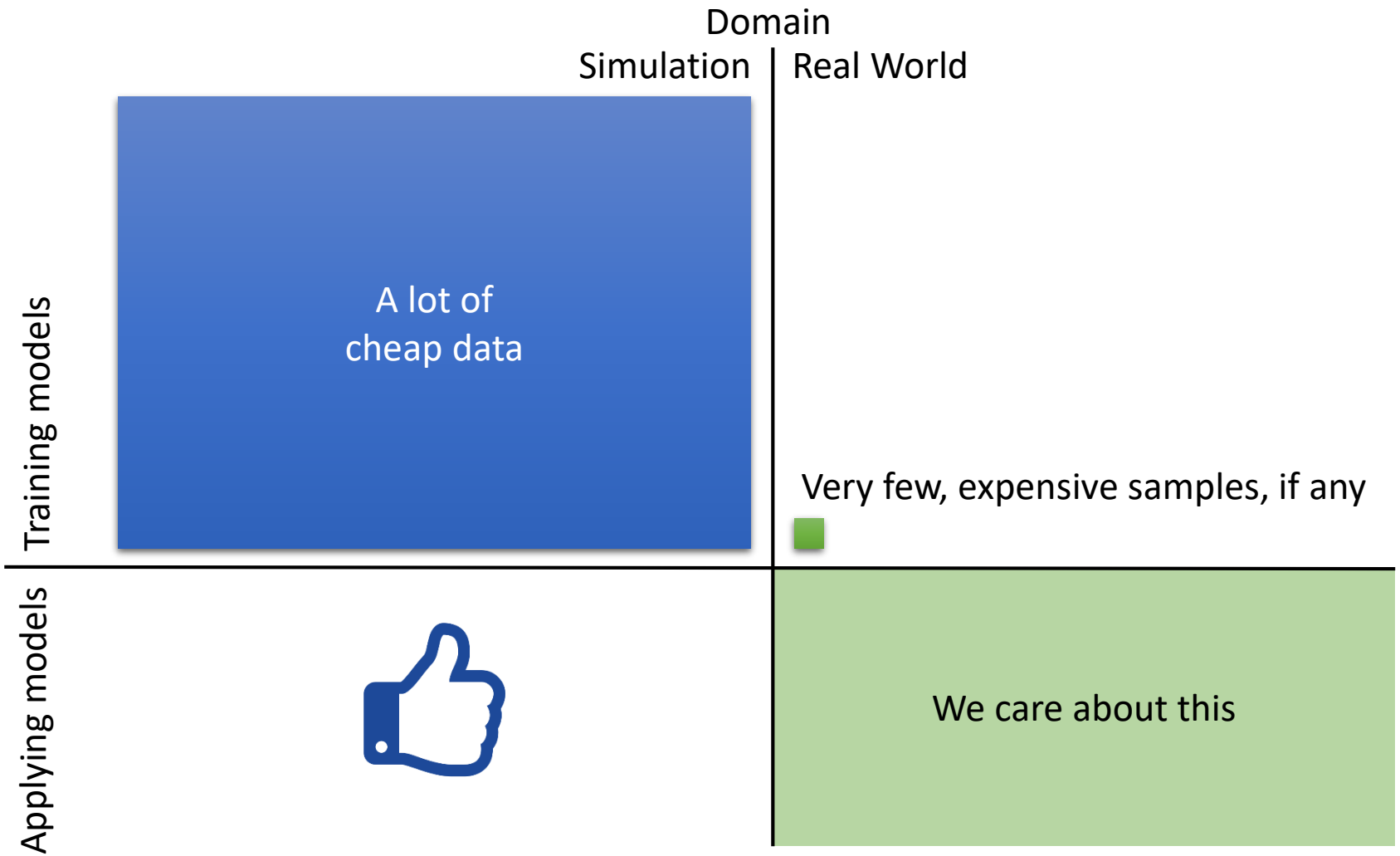
Computing a traversability map



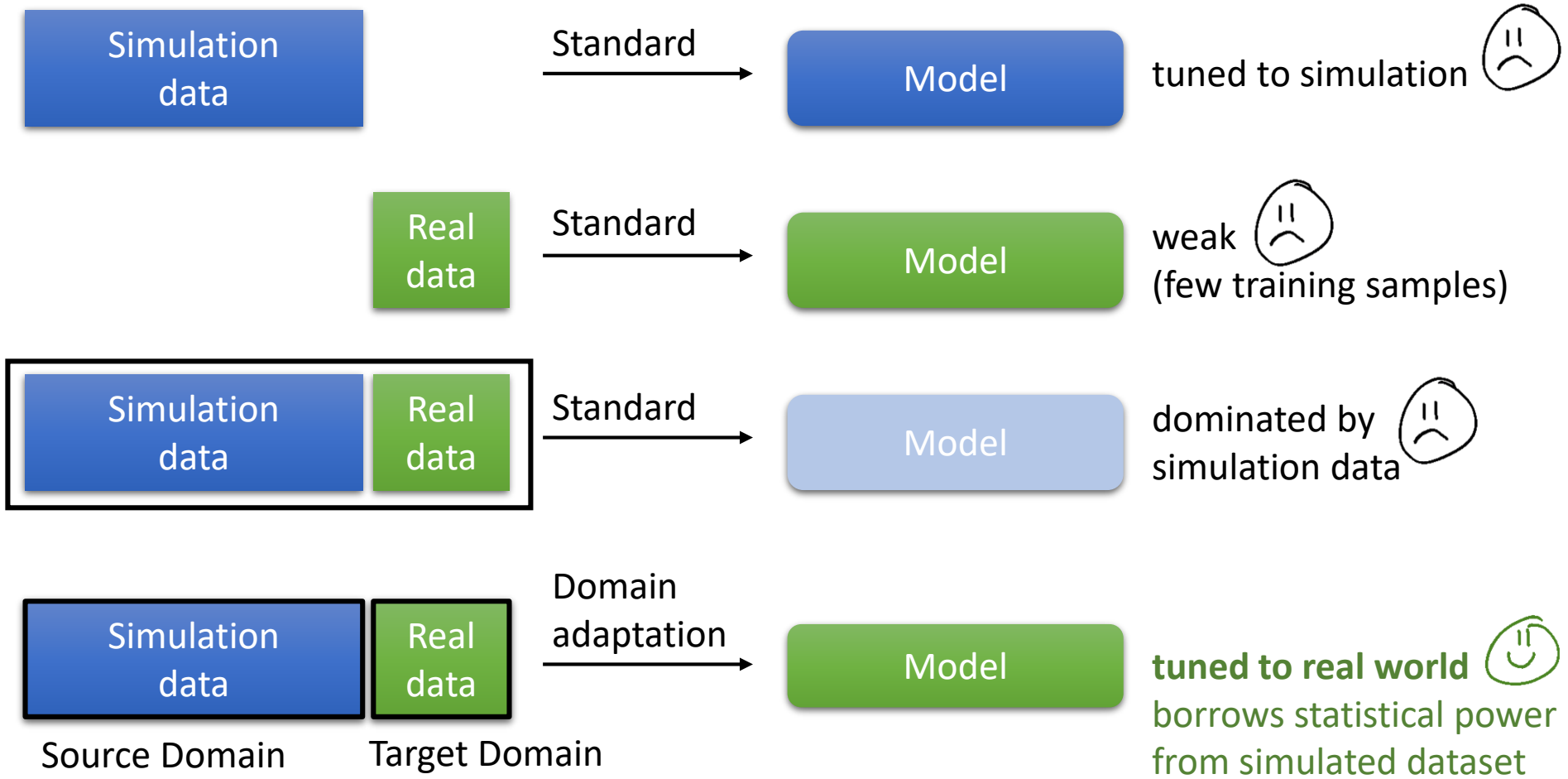
Oriented travrsability



How is this related to Domain Adaptation?



Domain adaptation: a scientific challenge



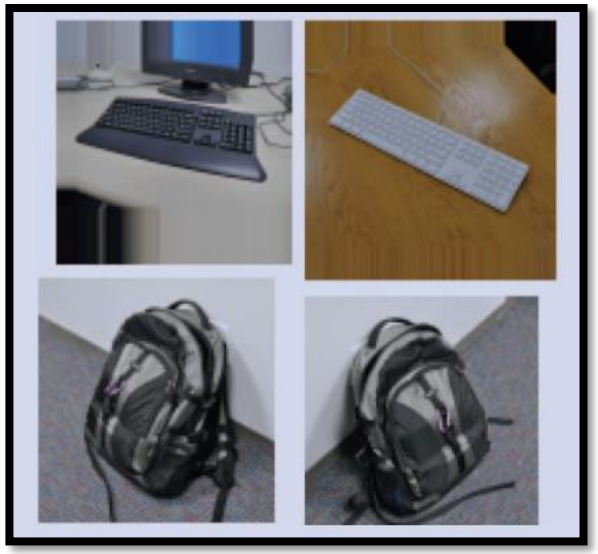
Real data comes from different sources (domains)

Office dataset (multi-domain object recognition benchmark)

Amazon

Caltech

DSLR



Webcam

2. Definitions and related problems

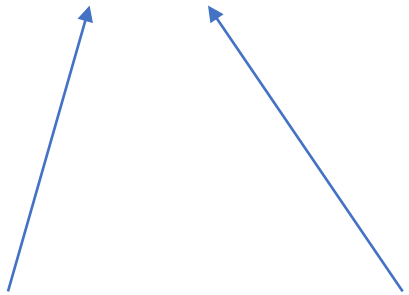
Main assumption in Domain Adaptation

Source domain

Target domain

Joint PDF

$$P_s(X,Y) \neq P_t(X,Y)$$



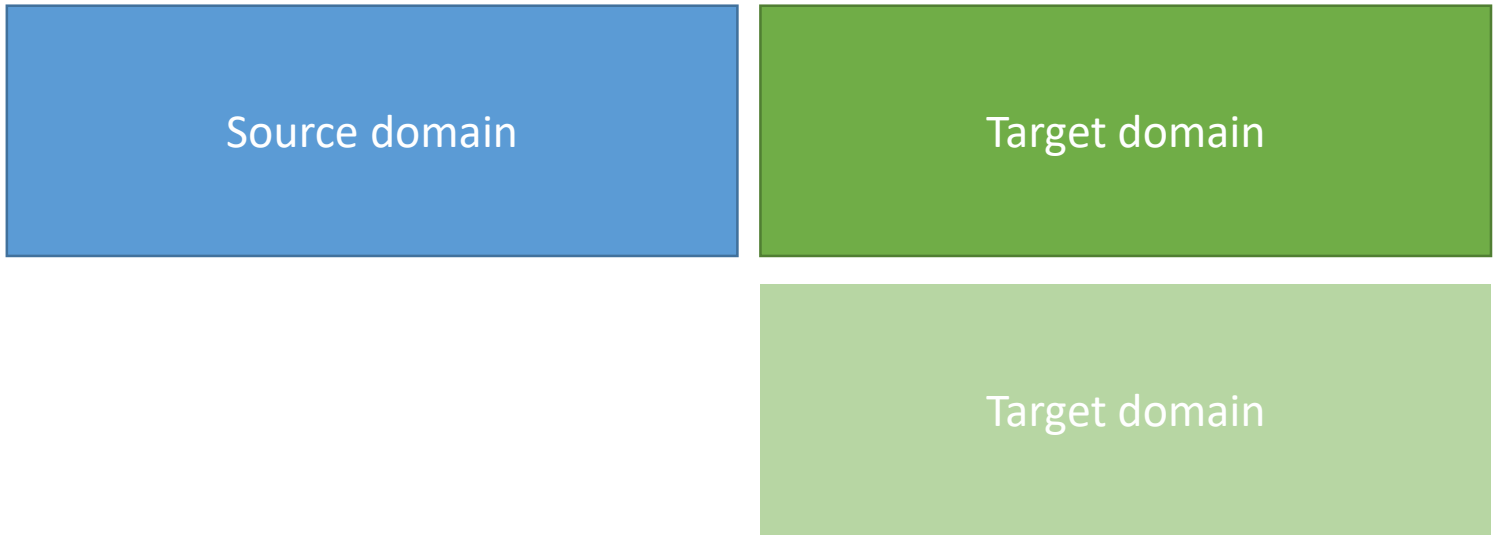
Random variable representing the input

Random variable representing the label

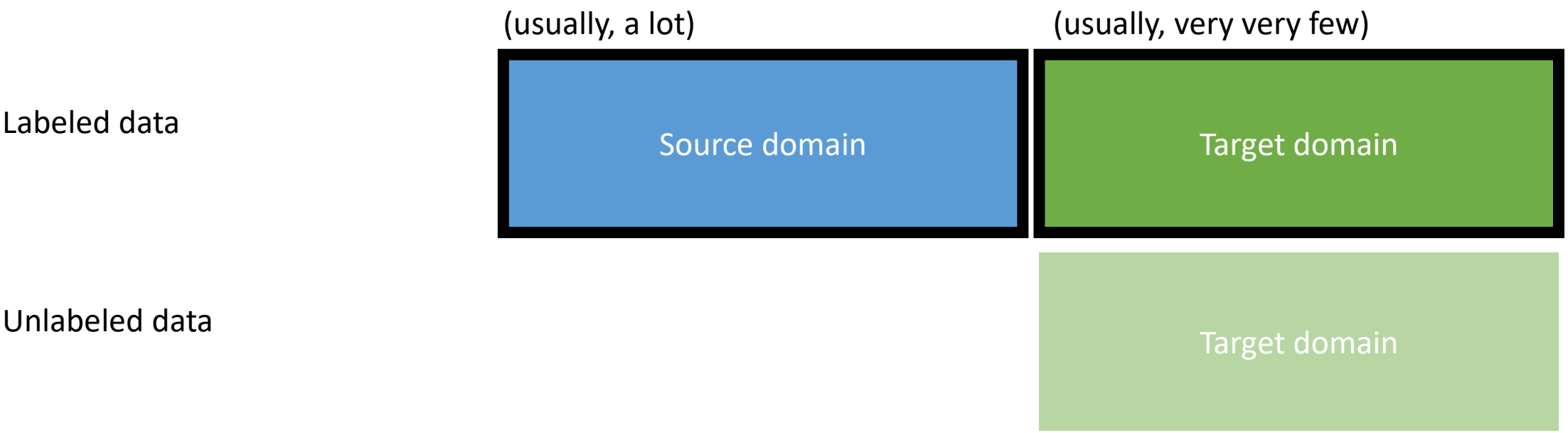
What training data do we have available?

Labeled data

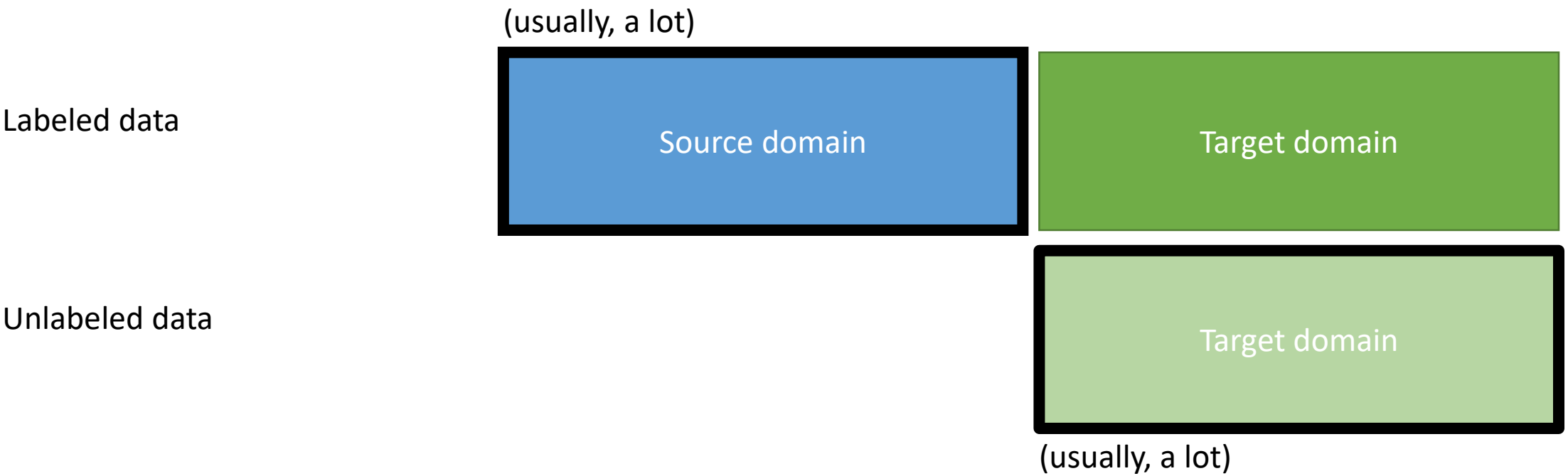
Unlabeled data



In semisupervised Domain Adaptation

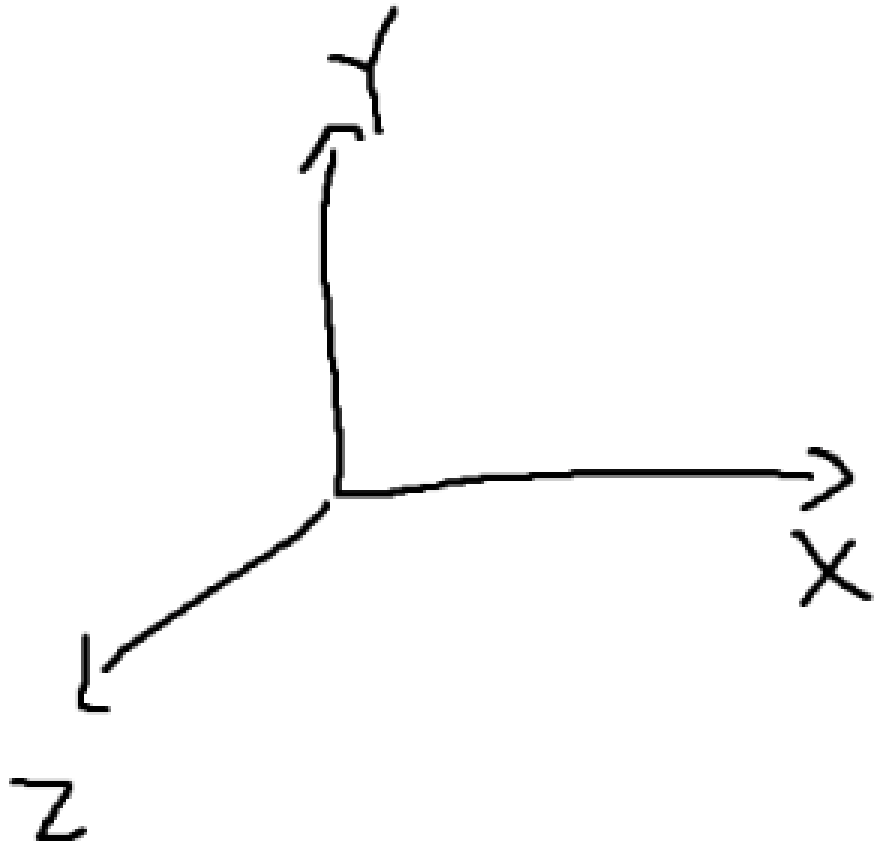


In unsupervised Domain Adaptation



Domain adaptation taxonomy

- Availability of labels in the target domain:
  - Yes: semisupervised DA
  - No: unsupervised DA
- Number of source domains:
  - One: (default)
  - Many: multisource DA
- Feature dimensionality among domains:
  - Same: (default)
  - Different: heterogeneous DA



Different but related problems

- Covariate shift

  Marginal distributions of X differ in the two domains
  but P(Y|X) is the same in the two domains

- Class imbalance

  Marginal distributions of Y differ in the two domains
  but P(X|Y) is the same in the two domains

- Transfer learning aka multitask learning

  Multiple tasks with different labels **but a single P(X) for all tasks**

- Semisupervised learning

  A single domain, with a mix of labeled and unlabeled data

- Self-taught learning aka self-supervised learning

  As above, but the unlabeled data might be very loosely related to the task (e.g. just share very simple features such as corners etc)

- Multiview learning aka cross-view or multimodal

  Samples from different domains are paired

3. Overview of some DA approaches

Frustratingly Easy Domain Adaptation

https://arxiv.org/abs/0907.1815, 2009

Augments the feature vectors

(semisupervised)

Frustratingly Easy Domain Adaptation

Hal Daumé III
School of Computing
University of Utah
Salt Lake City, Utah 84112
me@hal3.name

Abstract

We describe an approach to domain adaptation that is appropriate exactly in the case when one has enough "target" data to do slightly better than just using only "source" data. Our approach is incredibly simple, easy to implement as a preprocessing step (10 lines of Perl!) and outperforms state-of-the-art approaches on a range of datasets. Moreover, it is trivially extended to a multi-domain adaptation problem, where one has data from a variety of different domains.

1 Introduction

The task of domain adaptation is to develop learning algorithms that can be easily ported from one domain to another—say, from newswire to biomedical documents. This problem is particularly interesting in NLP because we are often in the situation that we have a large collection of labeled data in one "source" domain (say, newswire) but truly desire a model that performs well in a second "target" domain. The approach we present in this paper is based on the idea of transforming the domain adaptation learning problem into a standard supervised learning problem to which any standard algorithm may be applied (eg., maxent, SVMs, etc.). Our transformation is incredibly simple: we augment the feature space of both the source and target data and use the result as input to a standard learning algorithm.

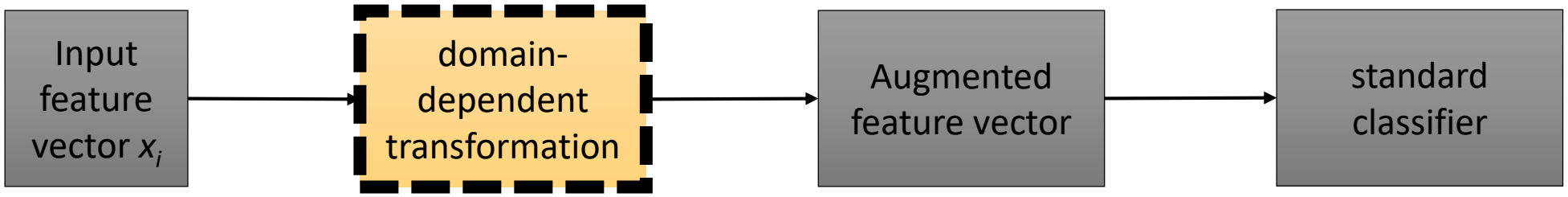
There are roughly two varieties of the domain adaptation problem that have been addressed in the literature: the fully supervised case and the semi-supervised case. The fully supervised case models the following scenario. We have access to a large, annotated corpus of data from a source domain. In addition, we spend a little money to annotate a small corpus in the target domain. We want to leverage both annotated datasets to obtain a model that performs well on the target domain. The semi-supervised case is similar, but instead of having a small annotated target corpus, we have a large but *unannotated* target corpus. In this paper, we focus exclusively on the fully supervised case.

One particularly nice property of our approach is that it is incredibly easy to implement: the Appendix provides a 10 line, 194 character Perl script for performing the complete transformation (available at `http://hal3.name/easyadapt.pl.gz`). In addition to this simplicity, our algorithm performs as well as (or, in some cases, better than) current state of the art techniques.

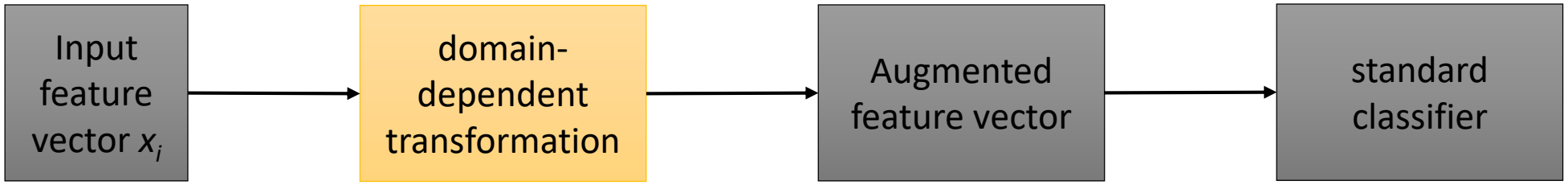
2 Problem Formalization and Prior Work

To facilitate discussion, we first introduce some notation. Denote by $\mathcal{X}$ the input space (typically either a real vector or a binary vector), and by $\mathcal{Y}$ the output space. We will write $\mathcal{D}^s$ to denote the distribution over source examples and $\mathcal{D}^t$ to denote the distribution over target examples. We assume access to a samples $D^s \sim \mathcal{D}^s$ of source examples from the source domain, and samples $D^t \sim \mathcal{D}^t$ of target examples from the target domain. We will assume that D^s is a collection of N examples and D^t is a collection of M examples (where, typically, $N \gg M$). Our goal is to learn a function $h : \mathcal{X} \to \mathcal{Y}$ with low expected loss with respect to the target domain.

Think about it!



How does it work?

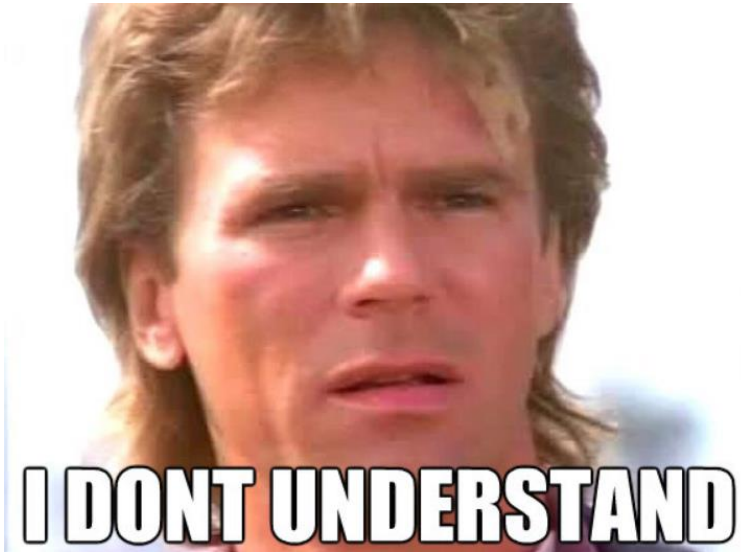


source domain

$$\Phi^s(\mathrm{x}_i^s) = \begin{bmatrix} \mathrm{x}_i^s \\ \mathrm{x}_i^s \\ 0_N \end{bmatrix}$$

target domain

$$\Phi^t(\mathrm{x}_i^{tl}) = \begin{bmatrix} \mathrm{x}_i^{tl} \\ 0_N \\ \mathrm{x}_i^{tl} \end{bmatrix}$$

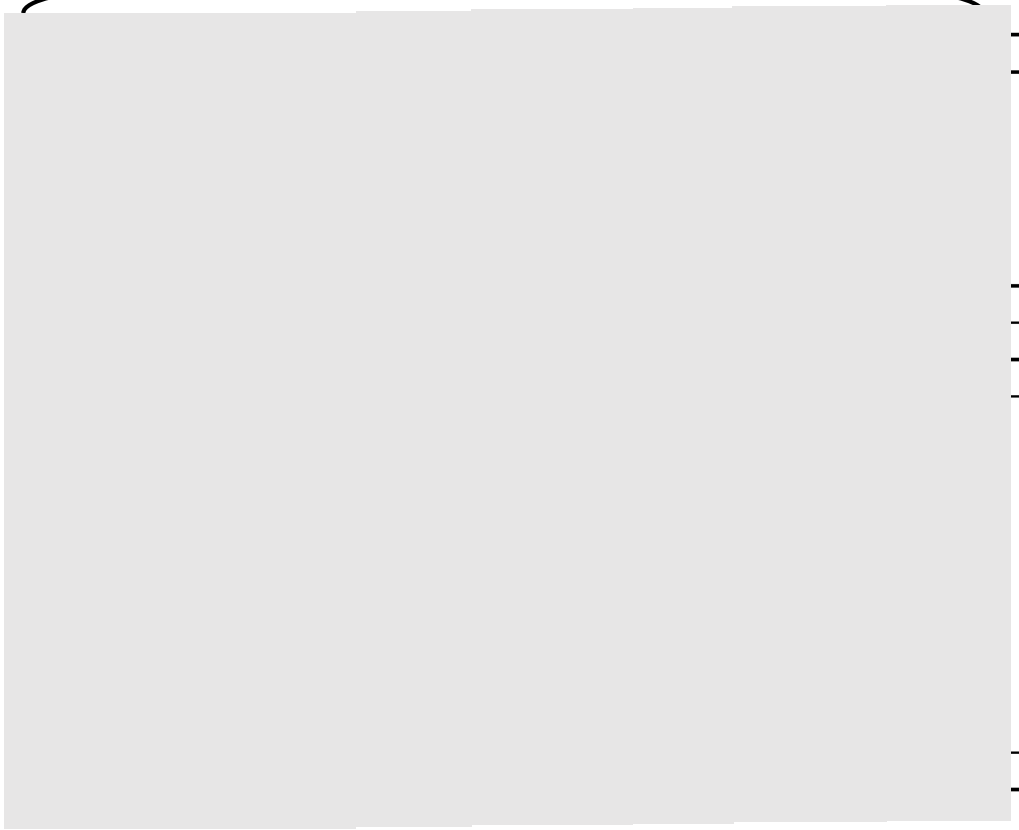


It actually works!

Classification errors on target domain, lower is better

Baselines

Datasets



AUGMENT
1.98
3.47
3.39
2.12
1.91
0.32
1.76
3.61
3.37
4.11
3.51
5.15
4.90
5.41
5.73
4.89
4.42
4.78
6.30
4.65

Interval: brainstorm about DA baselines!

- Which other “trivial” approaches can you think of that could be used as baselines in a Domain Adaptation setting?
- Brainstorm here!

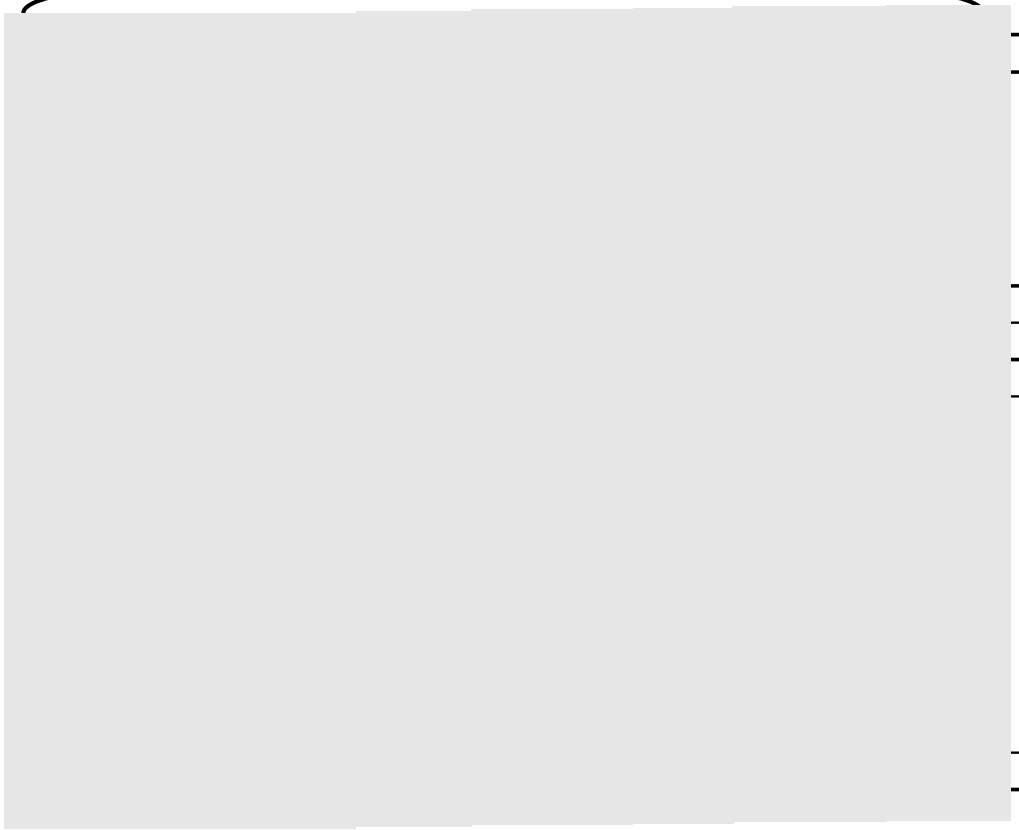
answergarden.ch/1279232

It actually works!

Classification errors on target domain, lower is better

Baselines

Datasets



AUGMENT
1.98
3.47
3.39
2.12
1.91
0.32
1.76
3.61
3.37
4.11
3.51
5.15
4.90
5.41
5.73
4.89
4.42
4.78
6.30
4.65

It actually works!

Classification errors on target domain, lower is better

Baselines

Datasets

SrcOnly	TgtOnly	All	Weight	Pred	LinInt	Prior	Augment
4.98	2.37	2.29	2.23	2.11	2.21	2.06	**1.98**
4.54	4.07	3.55	3.53	3.89	4.01	**3.47**	**3.47**
4.78	3.71	3.86	3.65	3.56	3.79	3.68	**3.39**
2.45	2.45	**2.12**	**2.12**	2.45	2.33	2.41	**2.12**
3.67	2.46	2.48	2.40	2.18	2.10	2.03	**1.91**
2.08	0.46	0.40	0.40	0.46	0.44	**0.34**	**0.32**
2.49	2.95	1.80	**1.75**	2.13	**1.77**	1.89	**1.76**
12.02	4.15	5.43	4.15	4.14	3.95	3.99	**3.61**
10.29	3.82	3.67	3.45	3.46	3.44	**3.35**	**3.37**
6.63	4.35	4.33	4.30	4.32	4.32	4.27	**4.11**
15.90	4.15	4.50	4.10	4.13	4.09	3.60	**3.51**
5.16	6.27	4.85	4.80	4.78	**4.72**	5.22	5.15
4.32	5.36	**4.16**	**4.15**	4.27	4.30	4.25	4.90
5.05	6.32	5.05	4.98	**5.01**	**5.05**	5.27	5.41
5.66	6.60	5.42	**5.39**	**5.39**	5.53	5.99	5.73
3.57	6.59	**3.14**	**3.11**	3.15	3.31	4.08	4.89
4.60	5.56	4.27	4.22	**4.20**	**4.19**	4.48	4.42
4.82	5.62	4.63	**4.57**	**4.55**	**4.55**	4.87	4.78
5.78	9.13	5.71	5.19	5.20	**5.15**	6.71	6.30
6.35	5.75	4.80	4.75	4.81	4.72	4.72	**4.65**

The SrcOnly baseline ignores the target data and trains a single model, only on the source data.

The TgtOnly baseline trains a single model only on the target data.

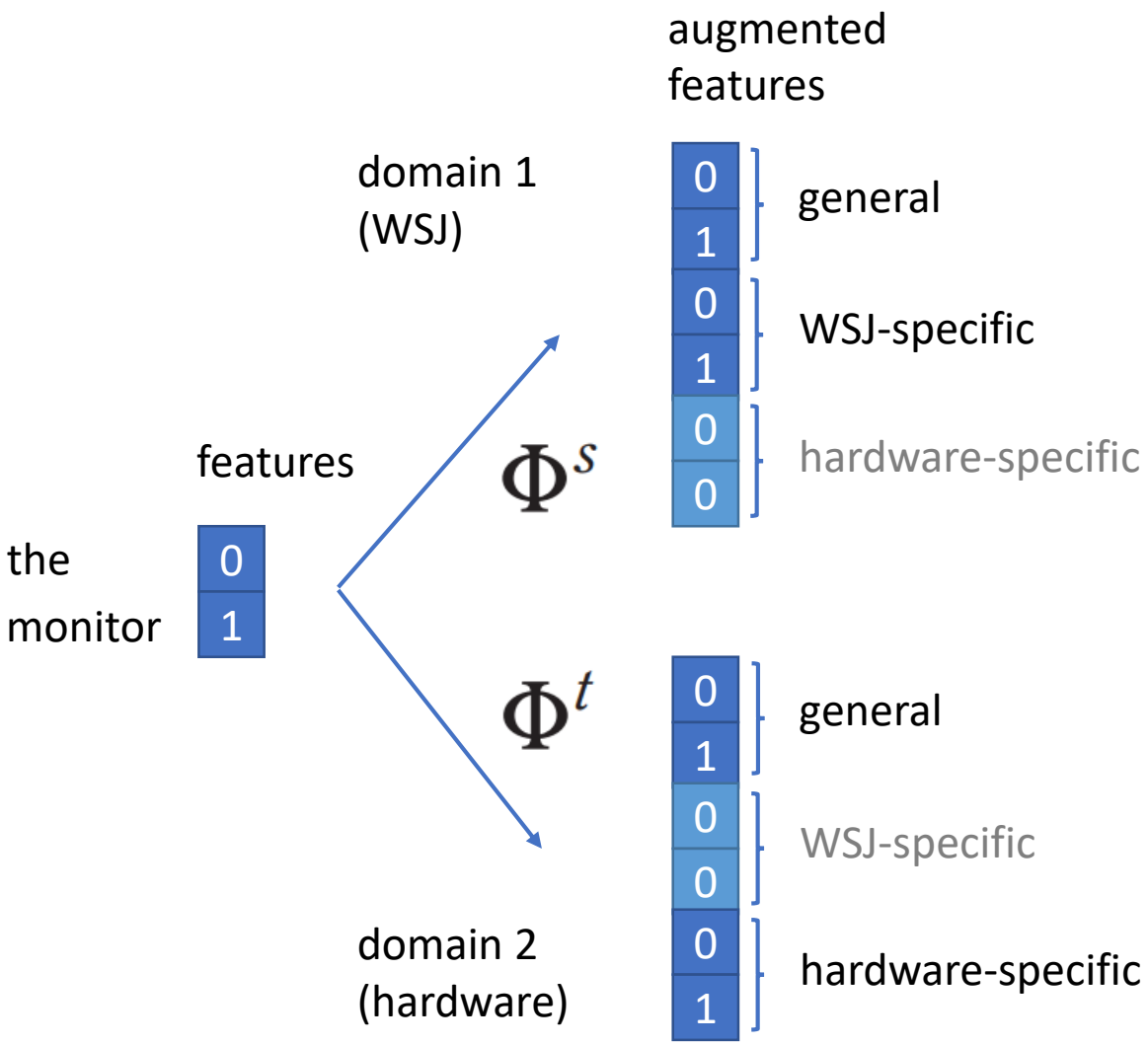
The All baseline simply trains a standard learning algorithm on the union of the two datasets.

A potential problem with the All baseline is that if $N \gg M$, then D^s may "wash out" any affect D^t might have. We will discuss this problem in more detail later, but one potential solution is to re-weight examples from D^s. For instance, if $N = 10 \times M$, we may weight each example from the source domain by 0.1. The next baseline, Weighted, is exactly this approach, with the weight chosen by cross-validation.

The Pred baseline is based on the idea of using the output of the source classifier as a feature in the target classifier. Specifically, we first train a SrcOnly model. Then we run the SrcOnly model on the target data (training, development and test). We use the predictions made by the SrcOnly model as additional features and train a second model on the target data, augmented with this new feature.

In the LinInt baseline, we linearly interpolate the predictions of the SrcOnly and the TgtOnly models. The interpolation parameter is adjusted based on target development data.

Intuition: **Why** does it work!?!



Before we proceed with a formal analysis of this transformation, let us consider why it might be expected to work. Suppose our task is part of speech tagging, our source domain is the Wall Street Journal and our target domain is a collection of reviews of computer hardware. Here, a word like "the" should be tagged as a determiner in both cases. However, a word like "monitor" is more likely to be a verb in the WSJ and more likely to be a noun in the hardware corpus. Consider a simple case where $\mathcal{X} = \mathbb{R}^2$, where x_1 indicates if the word is "the" and x_2 indicates if the word is "monitor." Then, in $\breve{\mathcal{X}}$, $\breve{x}_1$ and $\breve{x}_2$ will be "general" versions of the two indicator functions, $\breve{x}_3$ and $\breve{x}_4$ will be source-specific versions, and $\breve{x}_5$ and $\breve{x}_6$ will be target-specific versions.

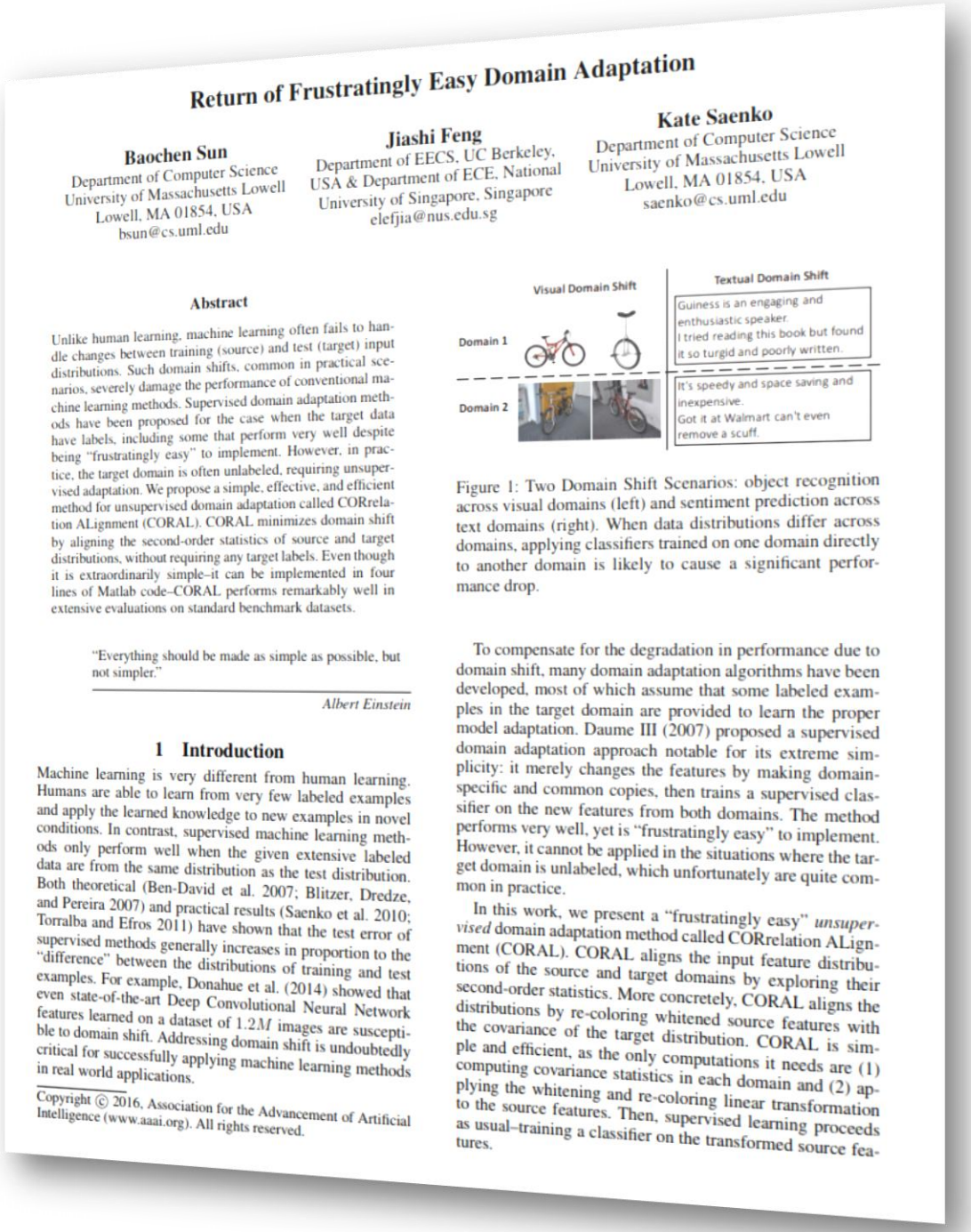
Now, consider what a learning algorithm could do to capture the fact that the appropriate tag for "the" remains constant across the domains, and the tag for "monitor" changes. In this case, the model can set the "determiner" weight vector to something like $\langle 1, 0, 0, 0, 0, 0 \rangle$. This places high weight on the common version of "the" and indicates that "the" is most likely a determiner, regardless of the domain. On the other hand, the weight vector for "noun" might look something like $\langle 0, 0, 0, 0, 0, 1 \rangle$, indicating that the word "monitor" is a noun *only* in the target domain. Similar, the weight vector for "verb" might look like $\langle 0, 0, 0, 1, 0, 0 \rangle$, indicating the "monitor" is a verb *only* in the source domain.

CORAL: Frustratingly Easy Domain Adaptation 2

http://www.aaai.org/ocs/index.php/AAAI/AAAI16/paper/download/12443/11842, AAAI 2016

Matches the distributions of the source and target features by **aligning the covariances**

(unsupervised)



Return of Frustratingly Easy Domain Adaptation

Baochen Sun
Department of Computer Science
University of Massachusetts Lowell
Lowell, MA 01854, USA
bsun@cs.uml.edu

Jiashi Feng
Department of EECS, UC Berkeley,
USA & Department of ECE, National
University of Singapore, Singapore
elefjia@nus.edu.sg

Kate Saenko
Department of Computer Science
University of Massachusetts Lowell
Lowell, MA 01854, USA
saenko@cs.uml.edu

Abstract

Unlike human learning, machine learning often fails to handle changes between training (source) and test (target) input distributions. Such domain shifts, common in practical scenarios, severely damage the performance of conventional machine learning methods. Supervised domain adaptation methods have been proposed for the case when the target data have labels, including some that perform very well despite being "frustratingly easy" to implement. However, in practice, the target domain is often unlabeled, requiring unsupervised adaptation. We propose a simple, effective, and efficient method for unsupervised domain adaptation called CORrelation ALignment (CORAL). CORAL minimizes domain shift by aligning the second-order statistics of source and target distributions, without requiring any target labels. Even though it is extraordinarily simple–it can be implemented in four lines of Matlab code–CORAL performs remarkably well in extensive evaluations on standard benchmark datasets.

"Everything should be made as simple as possible, but not simpler."

Albert Einstein

1 Introduction

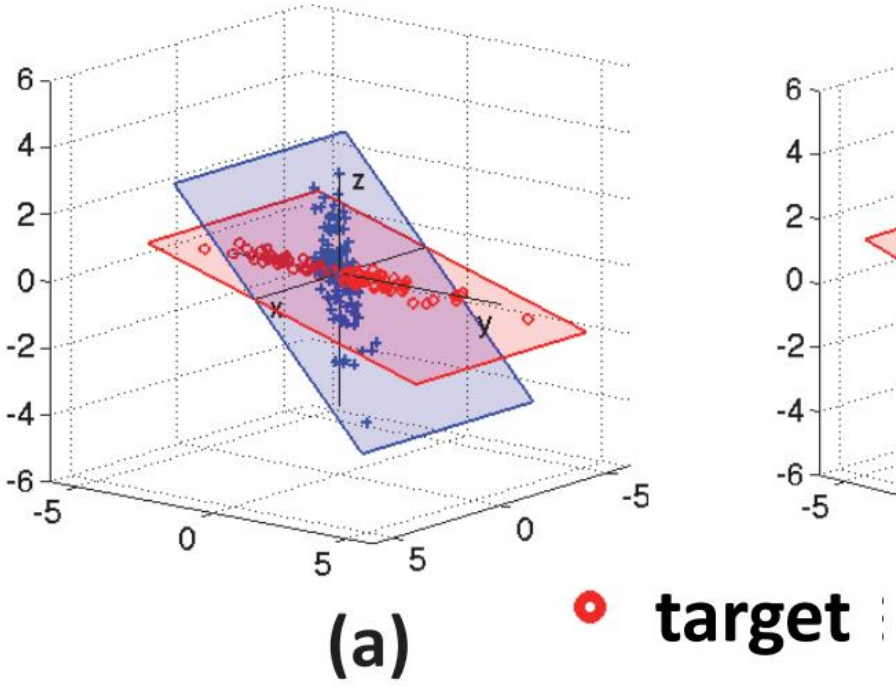
Machine learning is very different from human learning. Humans are able to learn from very few labeled examples and apply the learned knowledge to new examples in novel conditions. In contrast, supervised machine learning methods only perform well when the given extensive labeled data are from the same distribution as the test distribution. Both theoretical (Ben-David et al. 2007; Blitzer, Dredze, and Pereira 2007) and practical results (Saenko et al. 2010; Torralba and Efros 2011) have shown that the test error of supervised methods generally increases in proportion to the "difference" between the distributions of training and test examples. For example, Donahue et al. (2014) showed that even state-of-the-art Deep Convolutional Neural Network features learned on a dataset of $1.2M$ images are susceptible to domain shift. Addressing domain shift is undoubtedly critical for successfully applying machine learning methods in real world applications.

Figure 1: Two Domain Shift Scenarios: object recognition across visual domains (left) and sentiment prediction across text domains (right). When data distributions differ across domains, applying classifiers trained on one domain directly to another domain is likely to cause a significant performance drop.

To compensate for the degradation in performance due to domain shift, many domain adaptation algorithms have been developed, most of which assume that some labeled examples in the target domain are provided to learn the proper model adaptation. Daume III (2007) proposed a supervised domain adaptation approach notable for its extreme simplicity: it merely changes the features by making domain-specific and common copies, then trains a supervised classifier on the new features from both domains. The method performs very well, yet is "frustratingly easy" to implement. However, it cannot be applied in the situations where the target domain is unlabeled, which unfortunately are quite common in practice.

In this work, we present a "frustratingly easy" *unsupervised* domain adaptation method called CORrelation ALignment (CORAL). CORAL aligns the input feature distributions of the source and target domains by exploring their second-order statistics. More concretely, CORAL aligns the distributions by re-coloring whitened source features with the covariance of the target distribution. CORAL is simple and efficient, as the only computations it needs are (1) computing covariance statistics in each domain and (2) applying the whitening and re-coloring linear transformation to the source features. Then, supervised learning proceeds as usual–training a classifier on the transformed source features.

How does it work?



(a) original data

(b) remove correlations in source domain

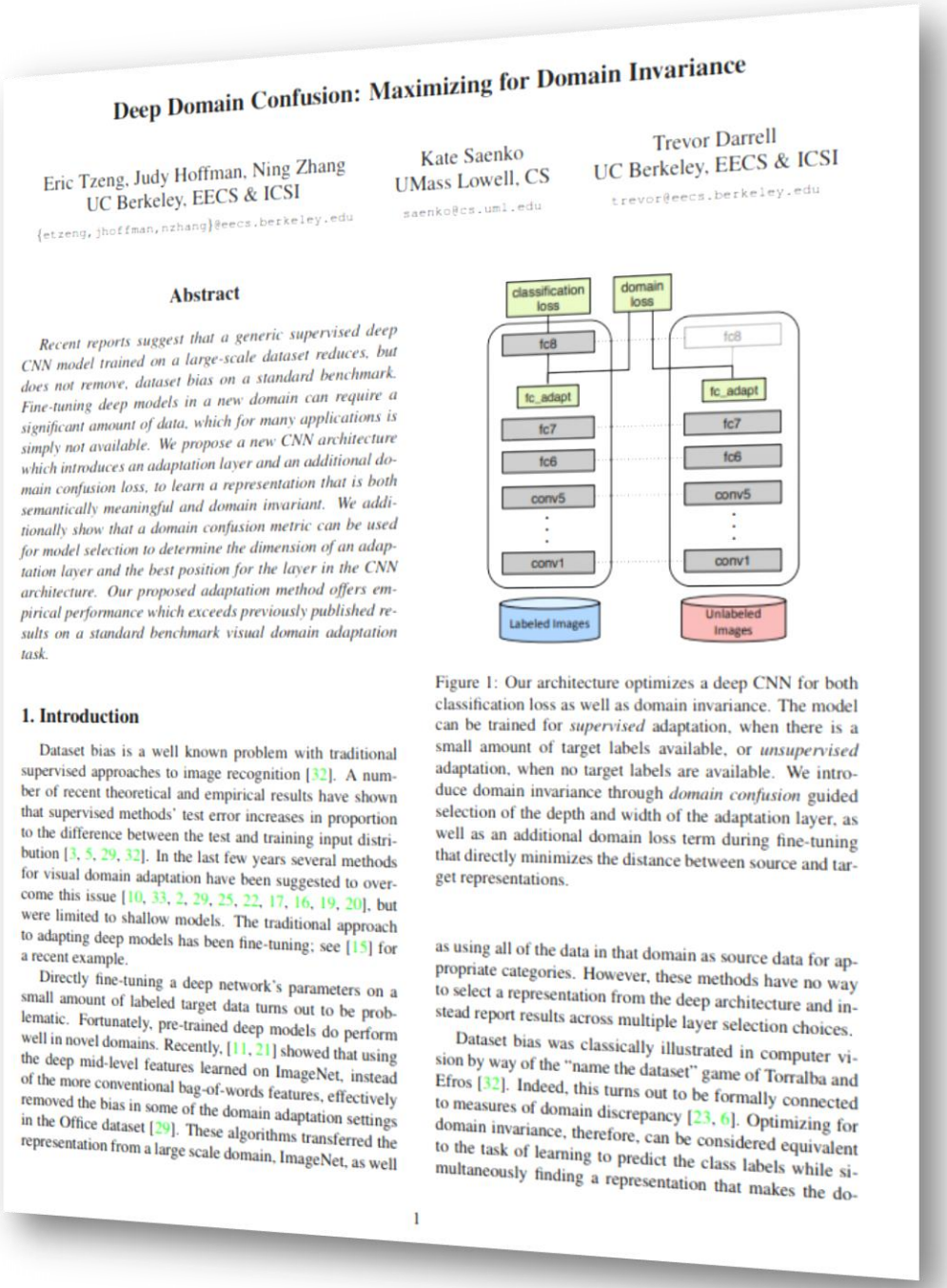
(c) copy feature correlations from target to source

Deep Domain Confusion

https://arxiv.org/abs/1412.3474, 2014

Introduces an adaptation layer and an additional domain confusion loss, to learn a representation that is both semantically meaningful and domain invariant

(unsupervised)



Deep Domain Confusion: Maximizing for Domain Invariance

Eric Tzeng, Judy Hoffman, Ning Zhang
UC Berkeley, EECS & ICSI
{etzeng,jhoffman,nzhang}@eecs.berkeley.edu

Kate Saenko
UMass Lowell, CS
saenko@cs.uml.edu

Trevor Darrell
UC Berkeley, EECS & ICSI
trevor@eecs.berkeley.edu

Abstract

Recent reports suggest that a generic supervised deep CNN model trained on a large-scale dataset reduces, but does not remove, dataset bias on a standard benchmark. Fine-tuning deep models in a new domain can require a significant amount of data, which for many applications is simply not available. We propose a new CNN architecture which introduces an adaptation layer and an additional domain confusion loss, to learn a representation that is both semantically meaningful and domain invariant. We additionally show that a domain confusion metric can be used for model selection to determine the dimension of an adaptation layer and the best position for the layer in the CNN architecture. Our proposed adaptation method offers empirical performance which exceeds previously published results on a standard benchmark visual domain adaptation task.

Figure 1: Our architecture optimizes a deep CNN for both classification loss as well as domain invariance. The model can be trained for *supervised* adaptation, when there is a small amount of target labels available, or *unsupervised* adaptation, when no target labels are available. We introduce domain invariance through *domain confusion* guided selection of the depth and width of the adaptation layer, as well as an additional domain loss term during fine-tuning that directly minimizes the distance between source and target representations.

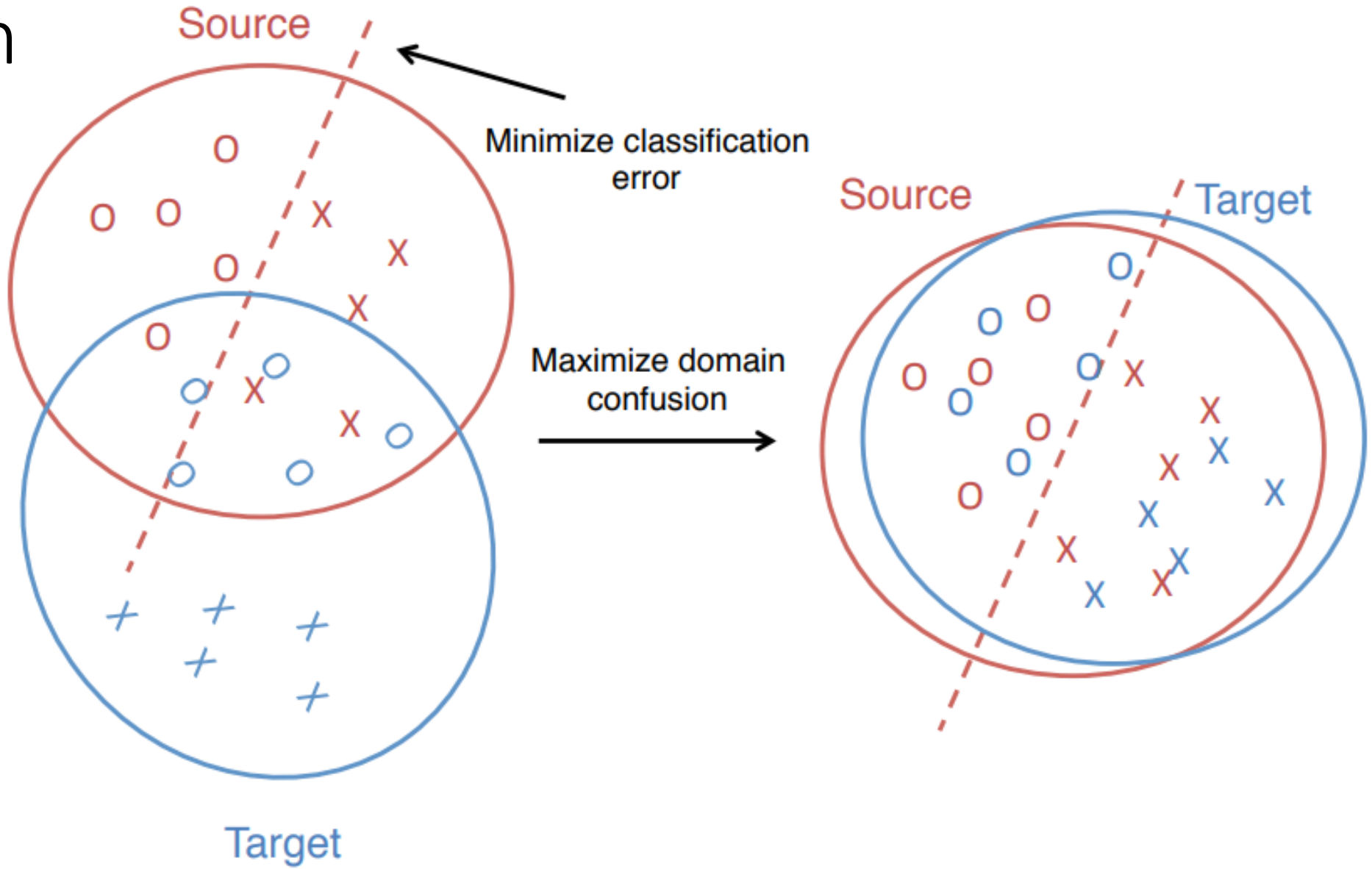
1. Introduction

Dataset bias is a well known problem with traditional supervised approaches to image recognition [32]. A number of recent theoretical and empirical results have shown that supervised methods' test error increases in proportion to the difference between the test and training input distribution [3, 5, 29, 32]. In the last few years several methods for visual domain adaptation have been suggested to overcome this issue [10, 33, 2, 29, 25, 22, 17, 16, 19, 20], but were limited to shallow models. The traditional approach to adapting deep models has been fine-tuning; see [15] for a recent example.

Directly fine-tuning a deep network's parameters on a small amount of labeled target data turns out to be problematic. Fortunately, pre-trained deep models do perform well in novel domains. Recently, [11, 21] showed that using the deep mid-level features learned on ImageNet, instead of the more conventional bag-of-words features, effectively removed the bias in some of the domain adaptation settings in the Office dataset [29]. These algorithms transferred the representation from a large scale domain, ImageNet, as well as using all of the data in that domain as source data for appropriate categories. However, these methods have no way to select a representation from the deep architecture and instead report results across multiple layer selection choices.

Dataset bias was classically illustrated in computer vision by way of the "name the dataset" game of Torralba and Efros [32]. Indeed, this turns out to be formally connected to measures of domain discrepancy [23, 6]. Optimizing for domain invariance, therefore, can be considered equivalent to the task of learning to predict the class labels while simultaneously finding a representation that makes the do-

Intuition

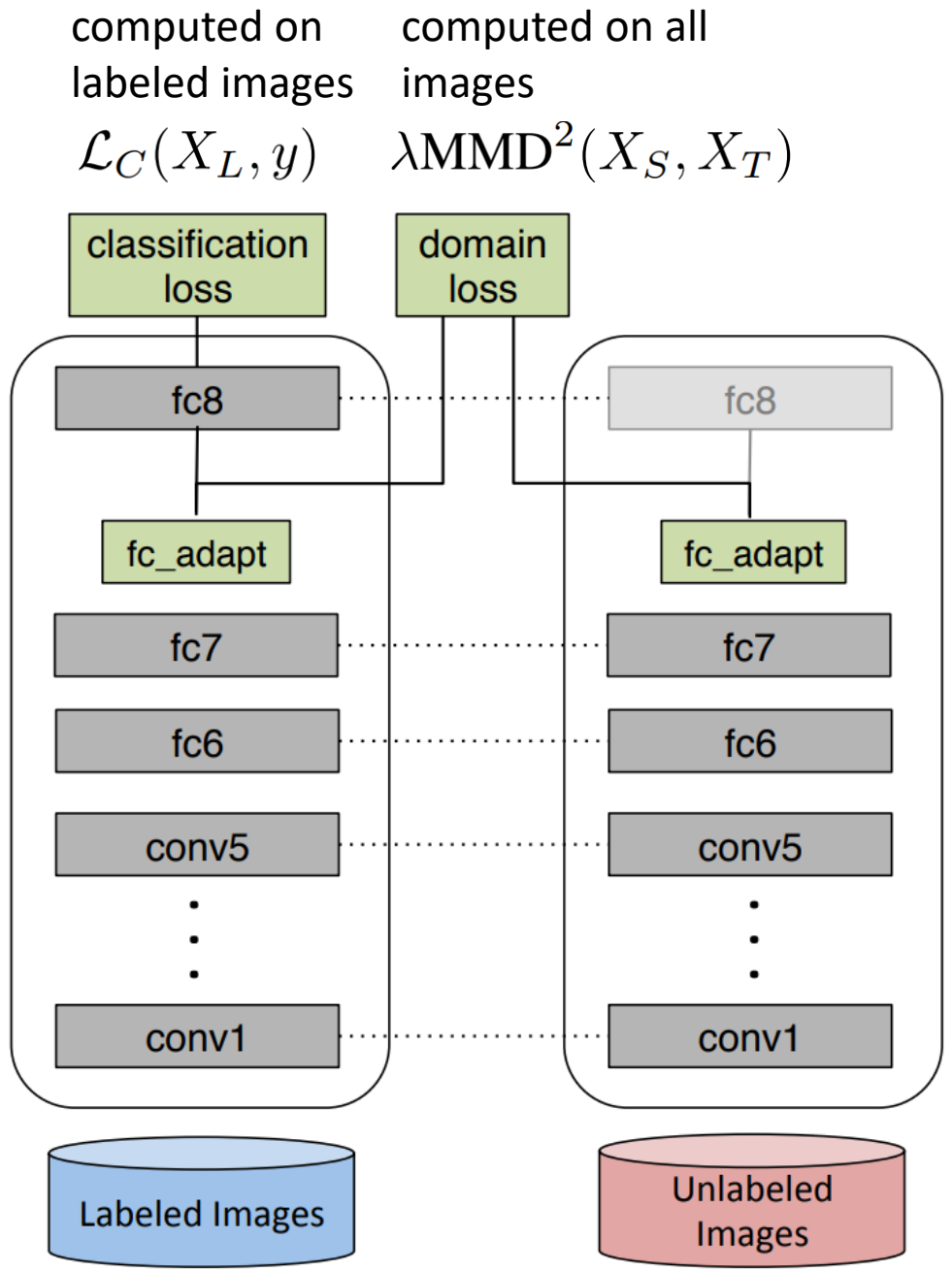

Source
Minimize classification error
Source
Target
Maximize domain confusion
Target

How does it work?

$$\mathcal{L} = \mathcal{L}_C(X_L, y) + \lambda \mathrm{MMD}^2(X_S, X_T)$$

MMD means Maximum Mean Discrepancy

$$\mathrm{MMD}(X_S, X_T) = \left\| \frac{1}{|X_S|} \sum_{x_s \in X_S} \phi(x_s) - \frac{1}{|X_T|} \sum_{x_t \in X_T} \phi(x_t) \right\|$$



Hyperparameter tuning by MMD minimization

Where should we put the adaptation layer?

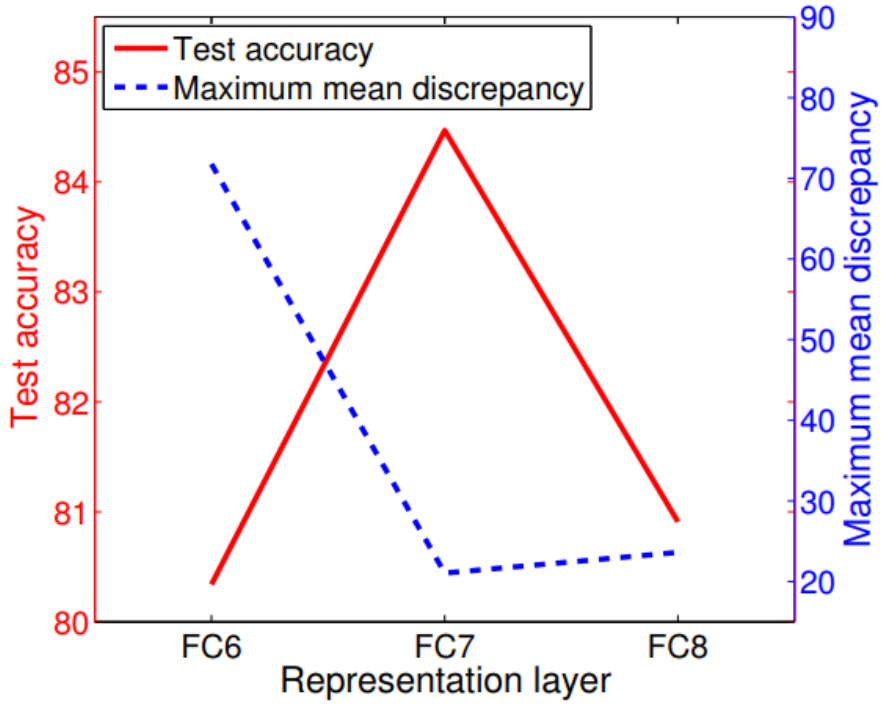


Figure 3: Maximum mean discrepancy and test accuracy for different choices of representation layer. We observe that MMD between source and target and accuracy on the target domain test set seem inversely related, indicating that MMD can be used to help select a layer for adaptation.

How large should it be?

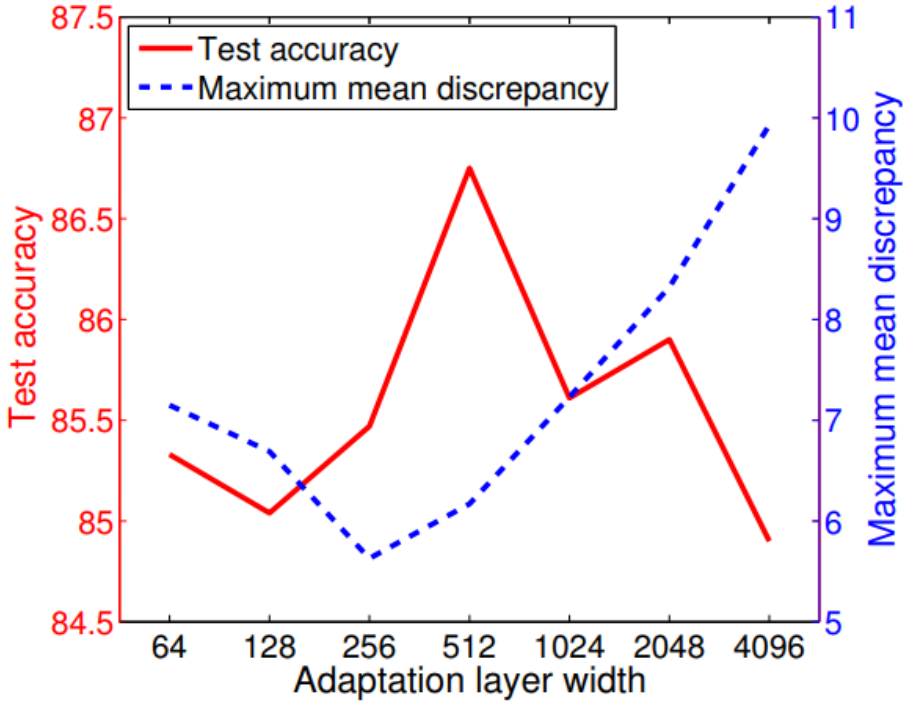
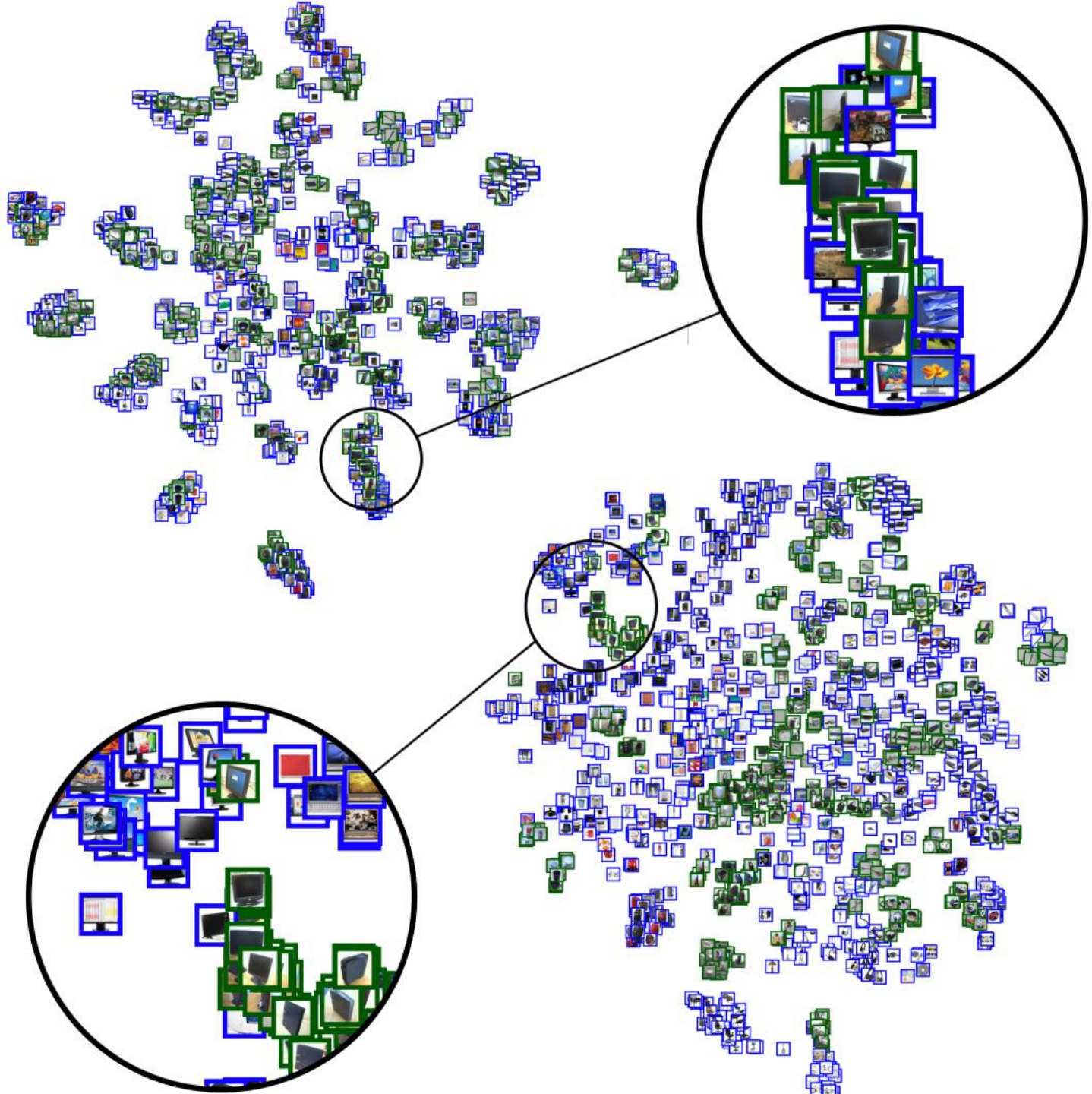


Figure 4: Maximum mean discrepancy and test accuracy for different values of adaptation layer dimensionality. We observe that MMD between source and target and accuracy on the target domain test set seem inversely related, indicating that MMD can be used to help select a dimensionality to use.

T-sne visualization of feature representation



Unsupervised Domain Adaptation by Backpropagation

2014, https://arxiv.org/abs/1409.7495

Promotes the emergence of features that are

(i) discriminative for the main learning task on the source domain and

(ii) invariant with respect to the shift between the domains

This adaptation behaviour can be achieved in almost any feed-forward model by augmenting it with few standard layers and a simple new gradient reversal layer. The resulting augmented architecture can be trained using standard backpropagation.

Unsupervised Domain Adaptation by Backpropagation

Yaroslav Ganin GANIN@SKOLTECH.RU
Victor Lempitsky LEMPITSKY@SKOLTECH.RU
Skolkovo Institute of Science and Technology (Skoltech)

Abstract

Top-performing deep architectures are trained on massive amounts of labeled data. In the absence of labeled data for a certain task, domain adaptation often provides an attractive option given that labeled data of similar nature but from a different domain (e.g. synthetic images) are available. Here, we propose a new approach to domain adaptation in deep architectures that can be trained on large amount of labeled data from the source domain and large amount of unlabeled data from the target domain (no labeled target-domain data is necessary).

As the training progresses, the approach promotes the emergence of "deep" features that are (i) discriminative for the main learning task on the source domain and (ii) invariant with respect to the shift between the domains. We show that this adaptation behaviour can be achieved in almost any feed-forward model by augmenting it with few standard layers and a simple new *gradient reversal* layer. The resulting augmented architecture can be trained using standard back-propagation.

Overall, the approach can be implemented with little effort using any of the deep-learning packages. The method performs very well in a series of image classification experiments, achieving adaptation effect in the presence of big domain shifts and outperforming previous state-of-the-art on Office datasets.

1. Introduction

Deep feed-forward architectures have brought impressive advances to the state-of-the-art across a wide variety of machine-learning tasks and applications. At the moment, however, these leaps in performance come only when a large amount of labeled training data is available. At the same time, for problems lacking labeled data, it may be still possible to obtain training sets that are big enough for training large-scale deep models, but that suffer from the *shift* in data distribution from the actual data encountered at "test time". One particularly important example is synthetic or semi-synthetic training data, which may come in abundance and be fully labeled, but which inevitably have a distribution that is different from real data (Liebelt & Schmid, 2010; Stark et al., 2010; Vázquez et al., 2014; Sun & Saenko, 2014).

Learning a discriminative classifier or other predictor in the presence of a *shift* between training and test distributions is known as *domain adaptation* (DA). A number of approaches to domain adaptation has been suggested in the context of *shallow* learning, e.g. in the situation when data representation/features are given and fixed. The proposed approaches then build the mappings between the *source* (training-time) and the *target* (test-time) domains, so that the classifier learned for the source domain can also be applied to the target domain, when composed with the learned mapping between domains. The appeal of the domain adaptation approaches is the ability to learn a mapping between domains in the situation when the target domain data are either fully unlabeled (*unsupervised domain annotation*) or have few labeled samples (*semi-supervised domain adaptation*). Below, we focus on the harder unsupervised case, although the proposed approach can be generalized to the semi-supervised case rather straightforwardly.

Unlike most previous papers on domain adaptation that worked with fixed feature representations, we focus on combining domain adaptation and deep feature learning within one training process (*deep domain adaptation*). Our goal is to embed domain adaptation into the process of learning representation, so that the final classification decisions are made based on features that are both discriminative and invariant to the change of domains, i.e. have the same or very similar distributions in the source and the target domains. In this way, the obtained feed-forward network can be applicable to the target domain without being hindered by the shift between the two domains.

We thus focus on learning features that combine (i) discriminativeness and (ii) domain-invariance. This is achieved by jointly optimizing the underlying features as well as two discriminative classifiers operating on these features: (i) the *label predictor* that predicts class labels and is used both during training and at test time and (ii) the

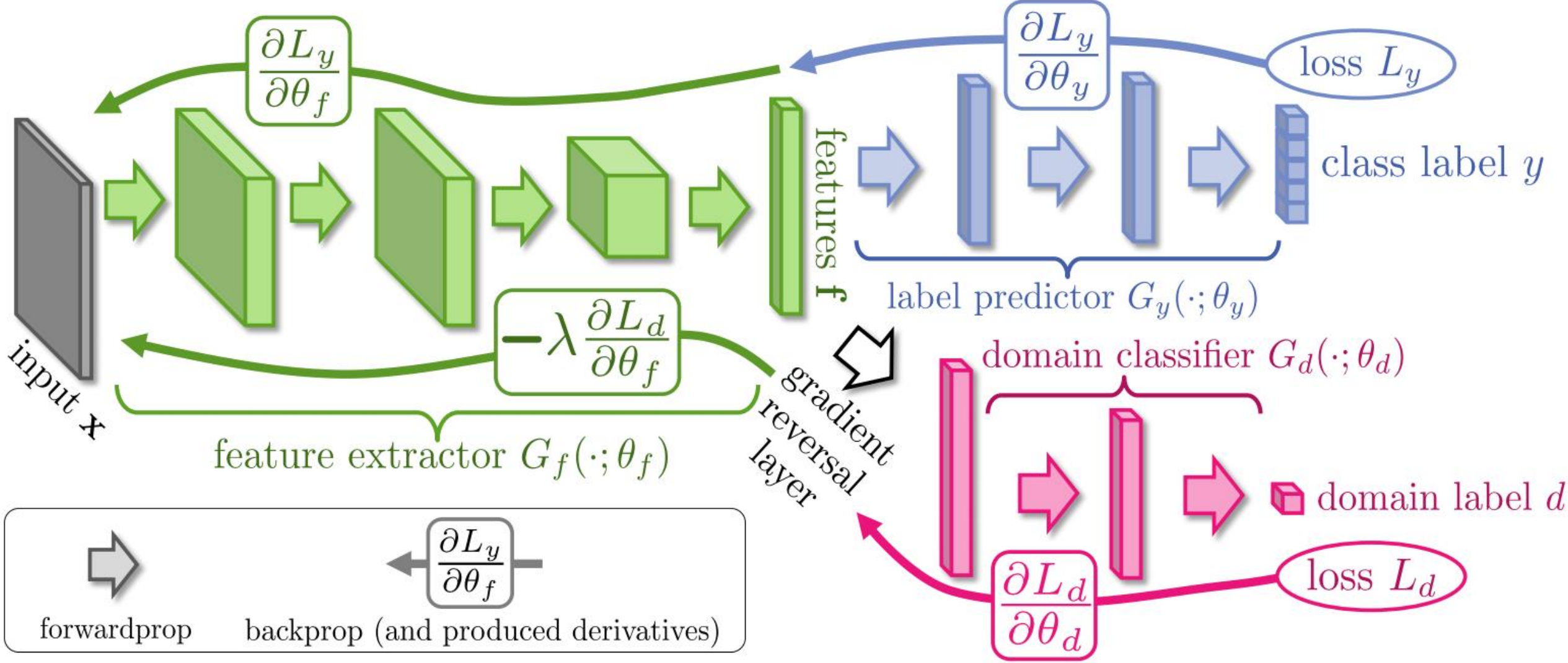


Figure 1. The **proposed architecture** includes a deep *feature extractor* (green) and a deep *label predictor* (blue), which together form a standard feed-forward architecture. Unsupervised domain adaptation is achieved by adding a *domain classifier* (red) connected to the feature extractor via a *gradient reversal layer* that multiplies the gradient by a certain negative constant during the backpropagation-based training. Otherwise, the training proceeds in a standard way and minimizes the label prediction loss (for source examples) and the domain classification loss (for all samples). Gradient reversal ensures that the feature distributions over the two domains are made similar (as indistinguishable as possible for the domain classifier), thus resulting in the domain-invariant features.

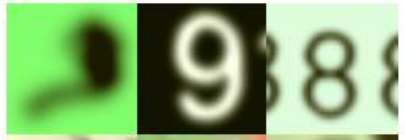
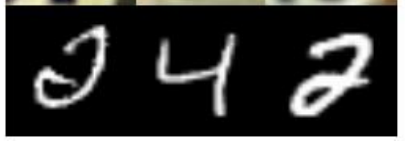
Principle

The parameters of both classifiers are optimized in order to minimize their error on the training set.

The parameters of the underlying deep feature mapping are optimized in order to:

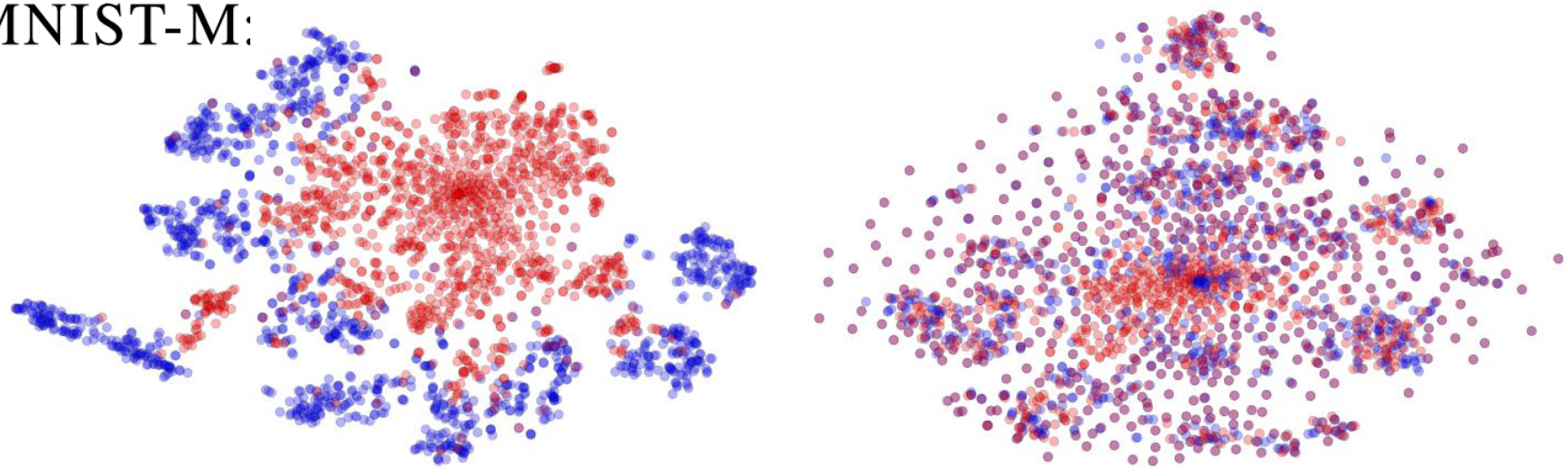
1) **minimize** the loss of the label classifier, and
2) to **maximize** (due to gradient reversal) the loss of the domain classifier. This encourages domain-invariant features to emerge in the course of the optimization.

Results & Datasets

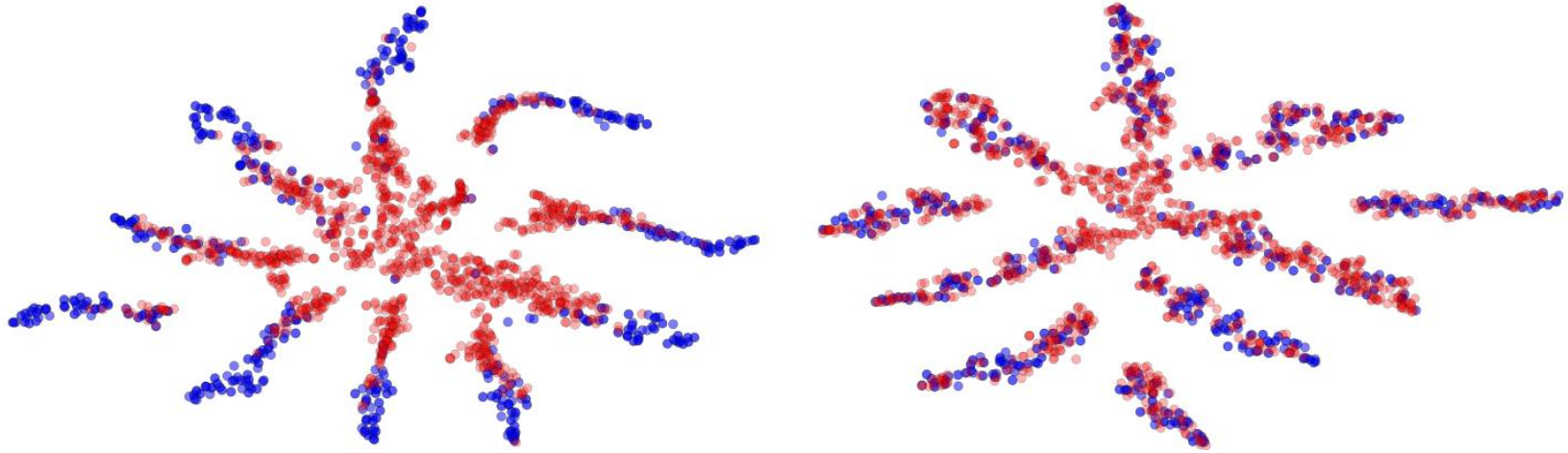
	MNIST	SYN NUMBERS	SVHN	SYN SIGNS
SOURCE				
TARGET				
	MNIST-M	SVHN	MNIST	GTSRB
SOURCE ONLY	.5749	.8665	.5919	.7400
SA (FERNANDO ET AL., 2013)	.6078 (7.9%)	.8672 (1.3%)	.6157 (5.9%)	.7635 (9.1%)
PROPOSED APPROACH	**.8149** (57.9%)	**.9048** (66.1%)	**.7107** (29.3%)	**.8866** (56.7%)
TRAIN ON TARGET	.9891	.9244	.9951	.9987

T-sne visualization of feature representation

MNIST → MNIST-M:



SYN NUMBERS → SVHN:

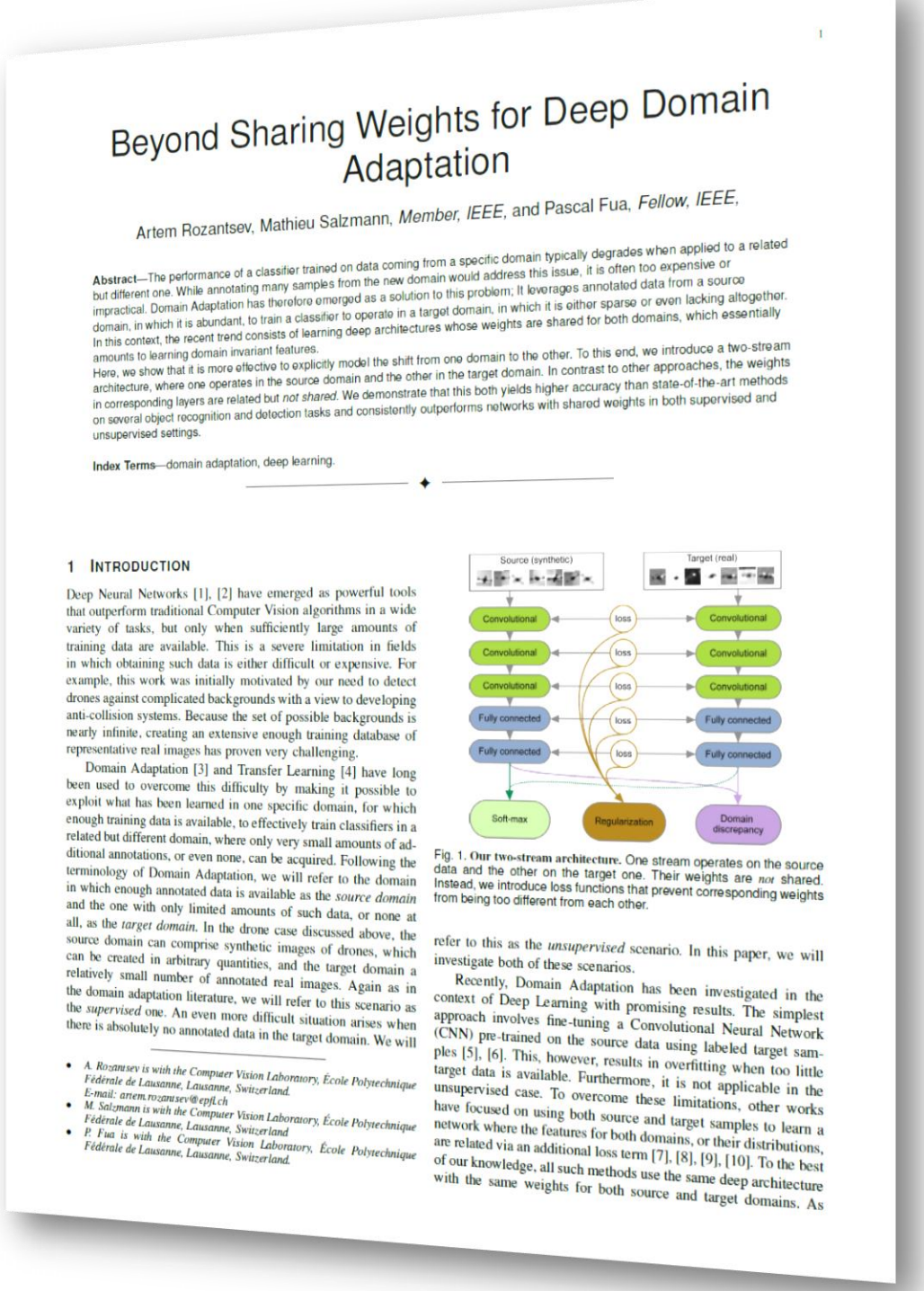


(a) Non-adapted (b) Adapted

Beyond Sharing Weights for Deep Domain Adaptation

2018
https://www.epfl.ch/labs/cvlab/research/domain-adaptation/deep-domain-adaptation/

- Maintains that Deep Learning approaches to Domain Adaptation should **not** focus on learning features that are invariant to the domain shift, which makes them less discriminative.
- Instead, explicitly models the domain shift using a two-stream CNN architecture, where the weights of the streams may or may not be shared. To encode the fact that both streams should be related, encourages the non-shared weights to remain close to being linear transformations of each other by introducing an additional loss term.



Beyond Sharing Weights for Deep Domain Adaptation

Artem Rozantsev, Mathieu Salzmann, *Member, IEEE*, and Pascal Fua, *Fellow, IEEE*,

Abstract—The performance of a classifier trained on data coming from a specific domain typically degrades when applied to a related but different one. While annotating many samples from the new domain would address this issue, it is often too expensive or impractical. Domain Adaptation has therefore emerged as a solution to this problem; It leverages annotated data from a source domain, in which it is abundant, to train a classifier to operate in a target domain, in which it is either sparse or even lacking altogether. In this context, the recent trend consists of learning deep architectures whose weights are shared for both domains, which essentially amounts to learning domain invariant features.
Here, we show that it is more effective to explicitly model the shift from one domain to the other. To this end, we introduce a two-stream architecture, where one operates in the source domain and the other in the target domain. In contrast to other approaches, the weights in corresponding layers are related but *not shared*. We demonstrate that this both yields higher accuracy than state-of-the-art methods on several object recognition and detection tasks and consistently outperforms networks with shared weights in both supervised and unsupervised settings.

Index Terms—domain adaptation, deep learning.

1 Introduction

Deep Neural Networks [1], [2] have emerged as powerful tools that outperform traditional Computer Vision algorithms in a wide variety of tasks, but only when sufficiently large amounts of training data are available. This is a severe limitation in fields in which obtaining such data is either difficult or expensive. For example, this work was initially motivated by our need to detect drones against complicated backgrounds with a view to developing anti-collision systems. Because the set of possible backgrounds is nearly infinite, creating an extensive enough training database of representative real images has proven very challenging.

Domain Adaptation [3] and Transfer Learning [4] have long been used to overcome this difficulty by making it possible to exploit what has been learned in one specific domain, for which enough training data is available, to effectively train classifiers in a related but different domain, where only very small amounts of additional annotations, or even none, can be acquired. Following the terminology of Domain Adaptation, we will refer to the domain in which enough annotated data is available as the *source domain* and the one with only limited amounts of such data, or none at all, as the *target domain*. In the drone case discussed above, the source domain can comprise synthetic images of drones, which can be created in arbitrary quantities, and the target domain a relatively small number of annotated real images. Again as in the domain adaptation literature, we will refer to this scenario as the *supervised* one. An even more difficult situation arises when there is absolutely no annotated data in the target domain. We will refer to this as the *unsupervised* scenario. In this paper, we will investigate both of these scenarios.

- A. Rozantsev is with the Computer Vision Laboratory, École Polytechnique Fédérale de Lausanne, Lausanne, Switzerland. E-mail: artem.rozantsev@epfl.ch
- M. Salzmann is with the Computer Vision Laboratory, École Polytechnique Fédérale de Lausanne, Lausanne, Switzerland
- P. Fua is with the Computer Vision Laboratory, École Polytechnique Fédérale de Lausanne, Lausanne, Switzerland.

Fig. 1. **Our two-stream architecture.** One stream operates on the source data and the other on the target one. Their weights are *not* shared. Instead, we introduce loss functions that prevent corresponding weights from being too different from each other.

Recently, Domain Adaptation has been investigated in the context of Deep Learning with promising results. The simplest approach involves fine-tuning a Convolutional Neural Network (CNN) pre-trained on the source data using labeled target samples [5], [6]. This, however, results in overfitting when too little target data is available. Furthermore, it is not applicable in the unsupervised case. To overcome these limitations, other works have focused on using both source and target samples to learn a network where the features for both domains, or their distributions, are related via an additional loss term [7], [8], [9], [10]. To the best of our knowledge, all such methods use the same deep architecture with the same weights for both source and target domains. As

Principle

The weights should be related, yet different for each of the two domains.

Allows the weights of the corresponding layers to differ between the two streams, but prevents them from being too far from each other. This models the fact that the source and target domains are related, and prevents overfitting in the target stream, when only very few labeled samples are available.

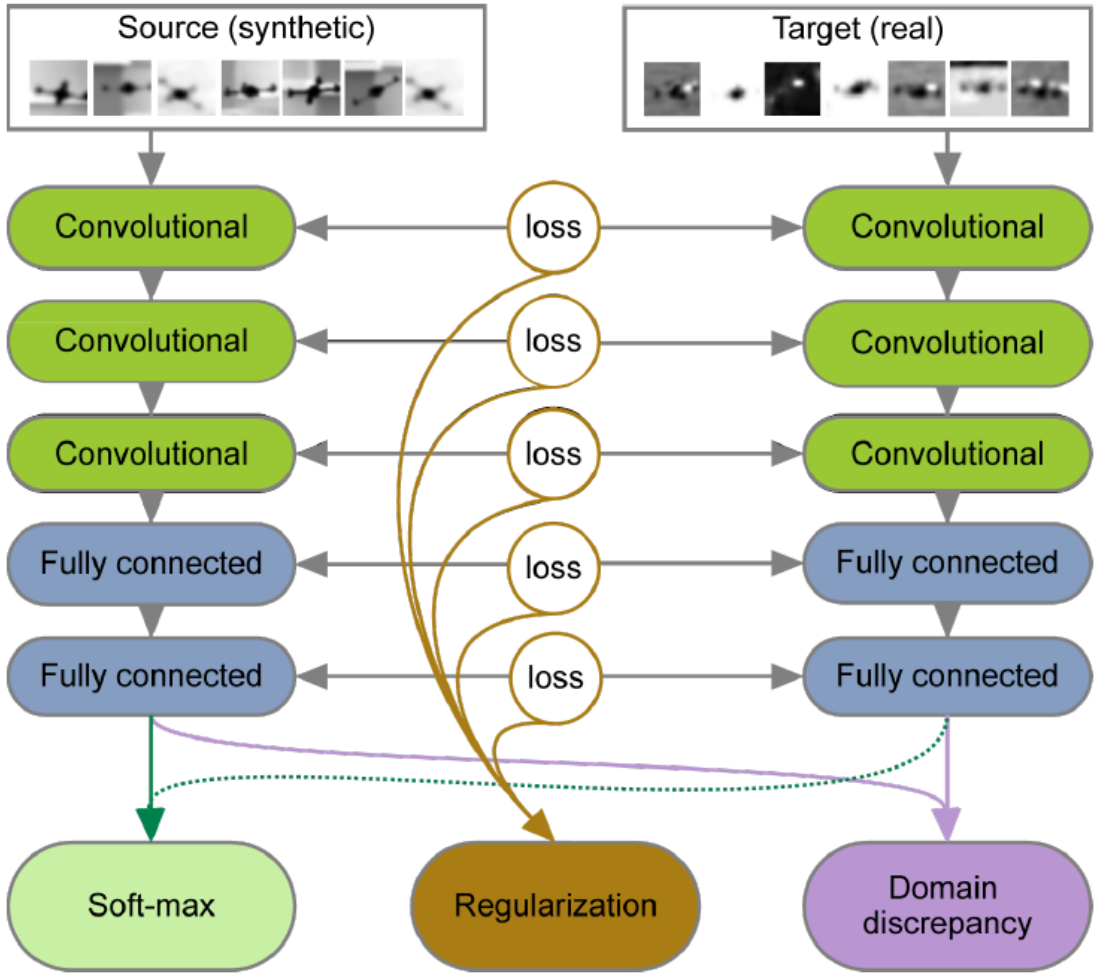
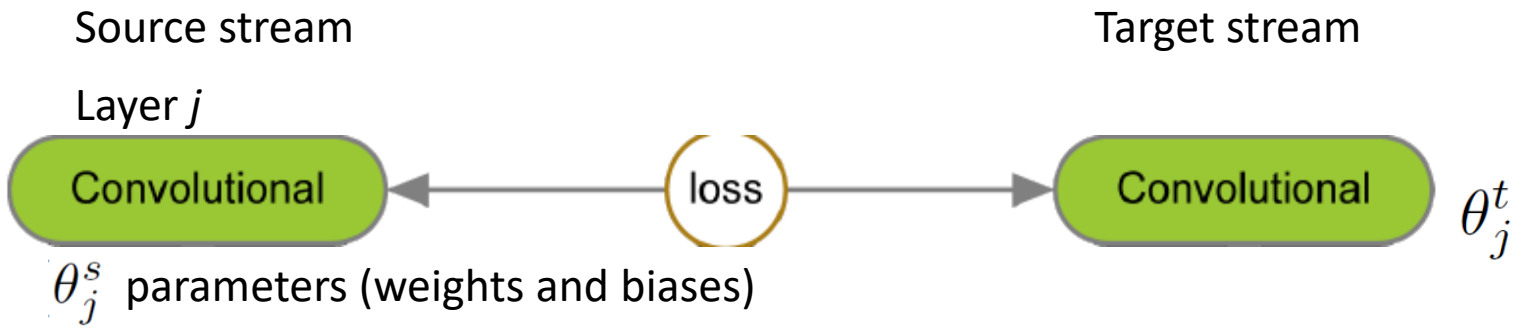


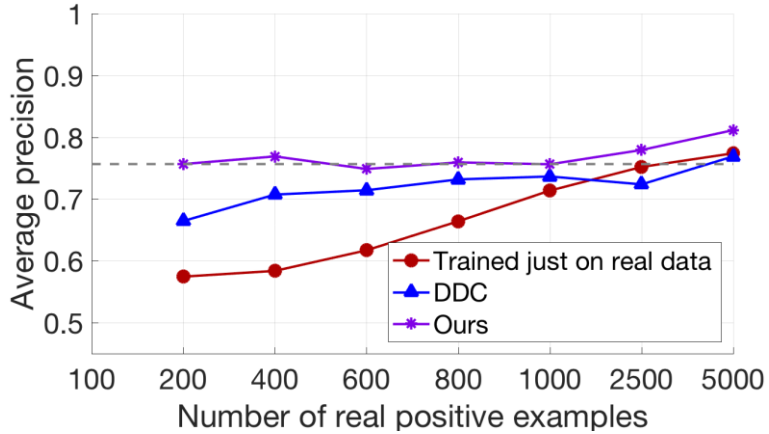
Fig. 1. **Our two-stream architecture.** One stream operates on the source data and the other on the target one. Their weights are *not* shared. Instead, we introduce loss functions that prevent corresponding weights from being too different from each other.

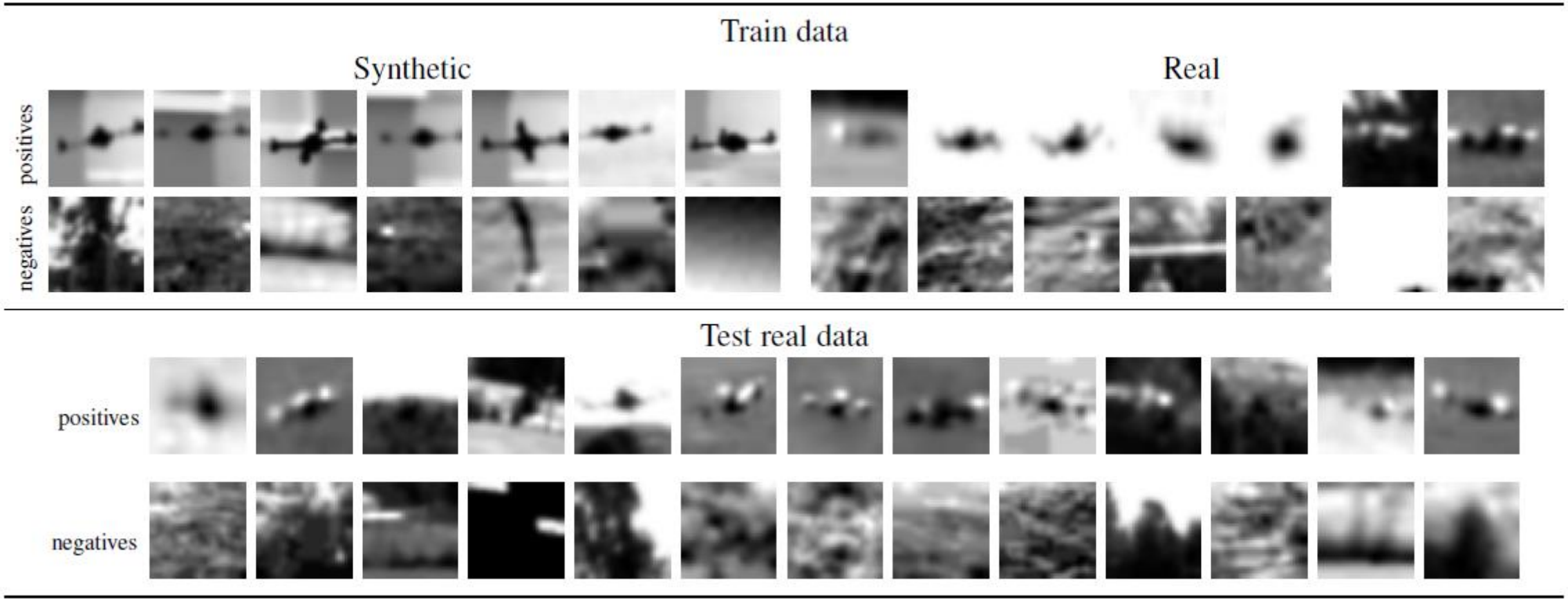
What's inside?



Application: detecting UAVS

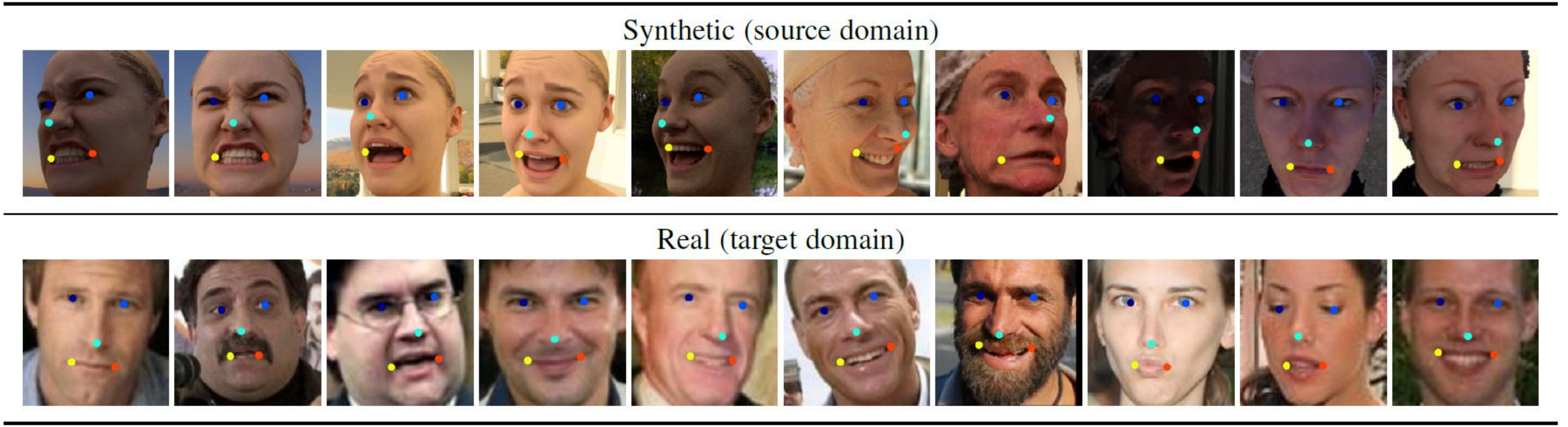
(semisupervised, classification)



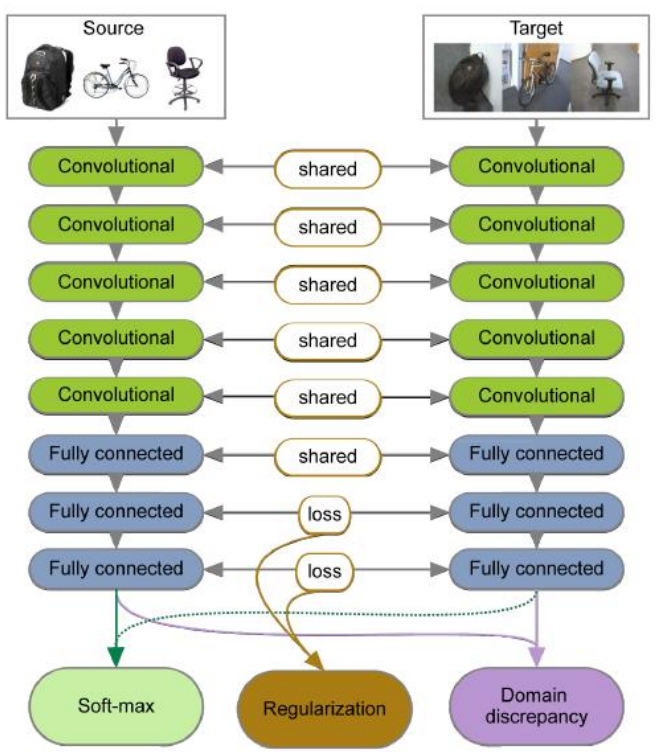


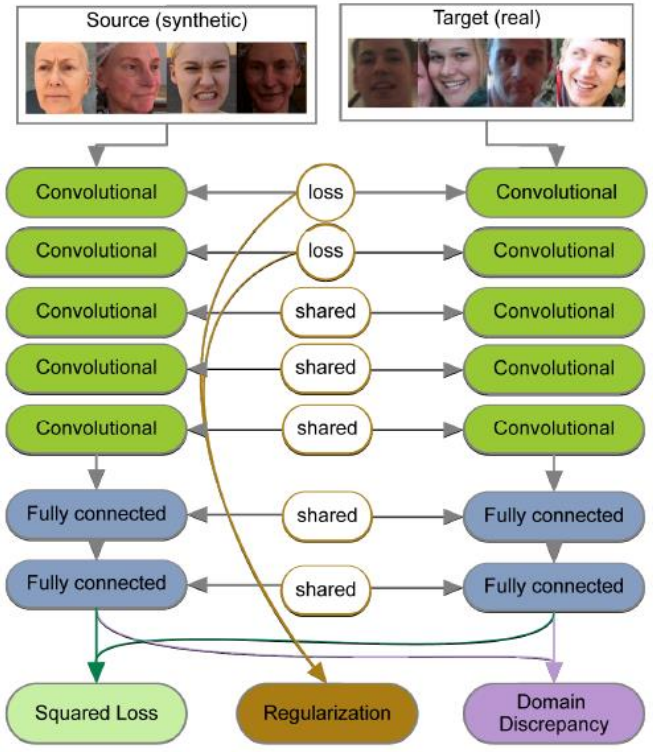
Application: face pose estimation

(unsupervised, regression)



Which layers should be shared? Depends on the task!



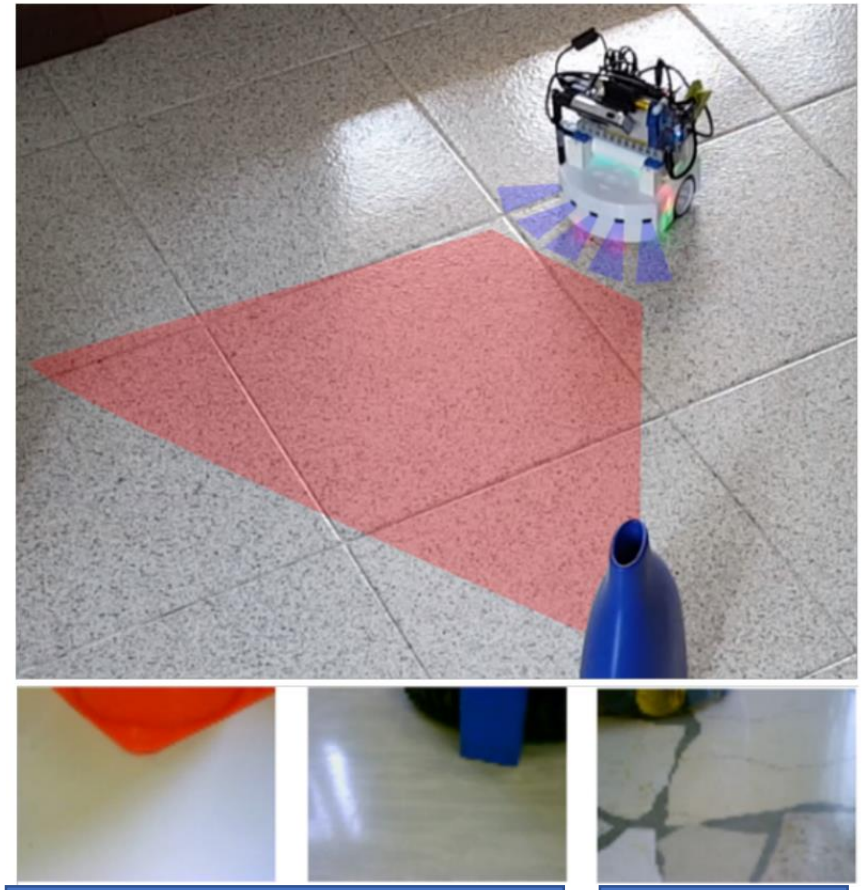


In all the experiments reported above, allowing the weights *not* to be shared in some fraction of the layers of our two-stream architecture boosts performance. This validates our initial hypothesis that explicitly modeling the domain shift is generally beneficial.

However, the optimal choice of which layers should or should not share their weights is application dependent. In the UAV detection and facial pose estimation cases allowing the weights in the first two layers to be different yields top performance, which we understand to mean that the domain shift is caused by low-level changes that are best handled in the early layers. By contrast, for the *Office* dataset, it is best to only allow the weights in the last two layers to differ. This network configuration was determined using *Amazon* and *Webcam* images, such as those shown in Fig. 8. Close examination of these images reveals that the differences between them are not simply due to low-level phenomena, such as illumination changes, but to more complex variations. It therefore seems reasonable that the higher layers of the network, which encode higher-level information, should be domain-specific.

4. Some first-hand experiences with Domain Adaptation

Generalize across domains for our Mighty Thymio

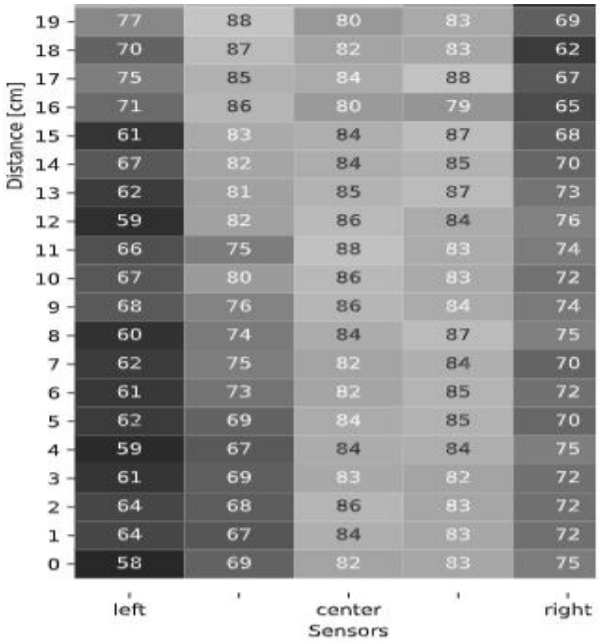


Source domains

Target

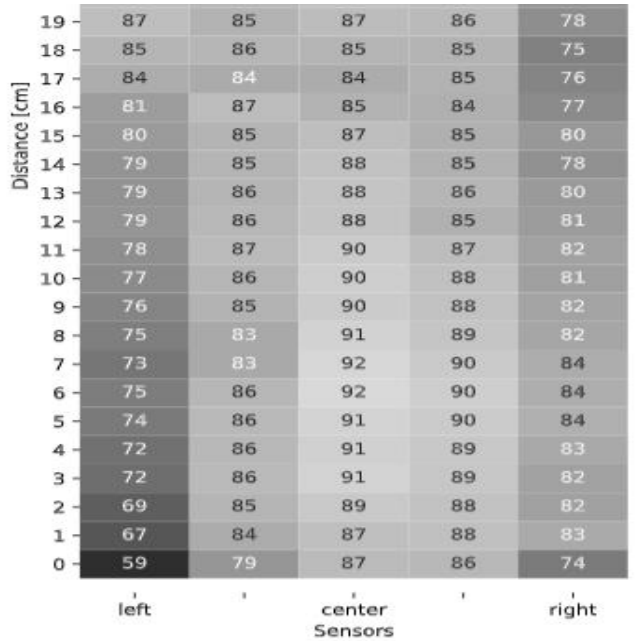
Quantitative obstacle detection performance

Without DA

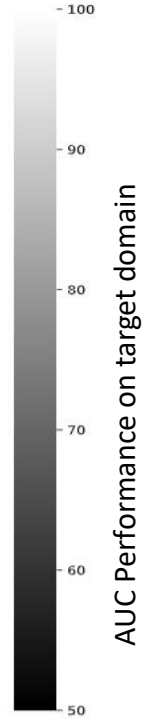


Baseline

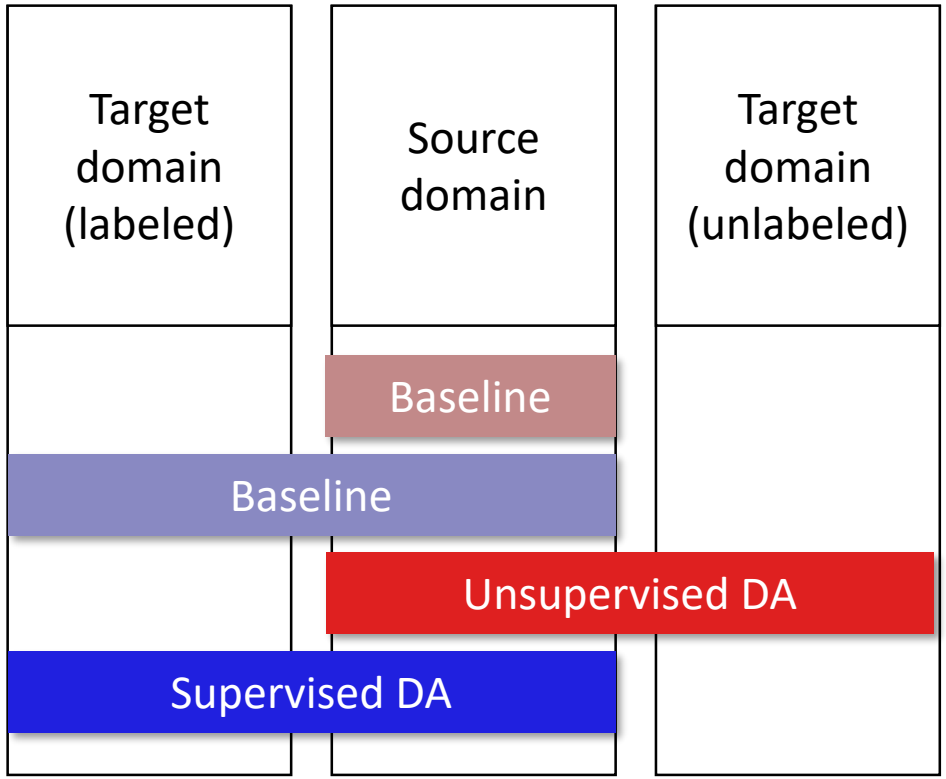
With DA

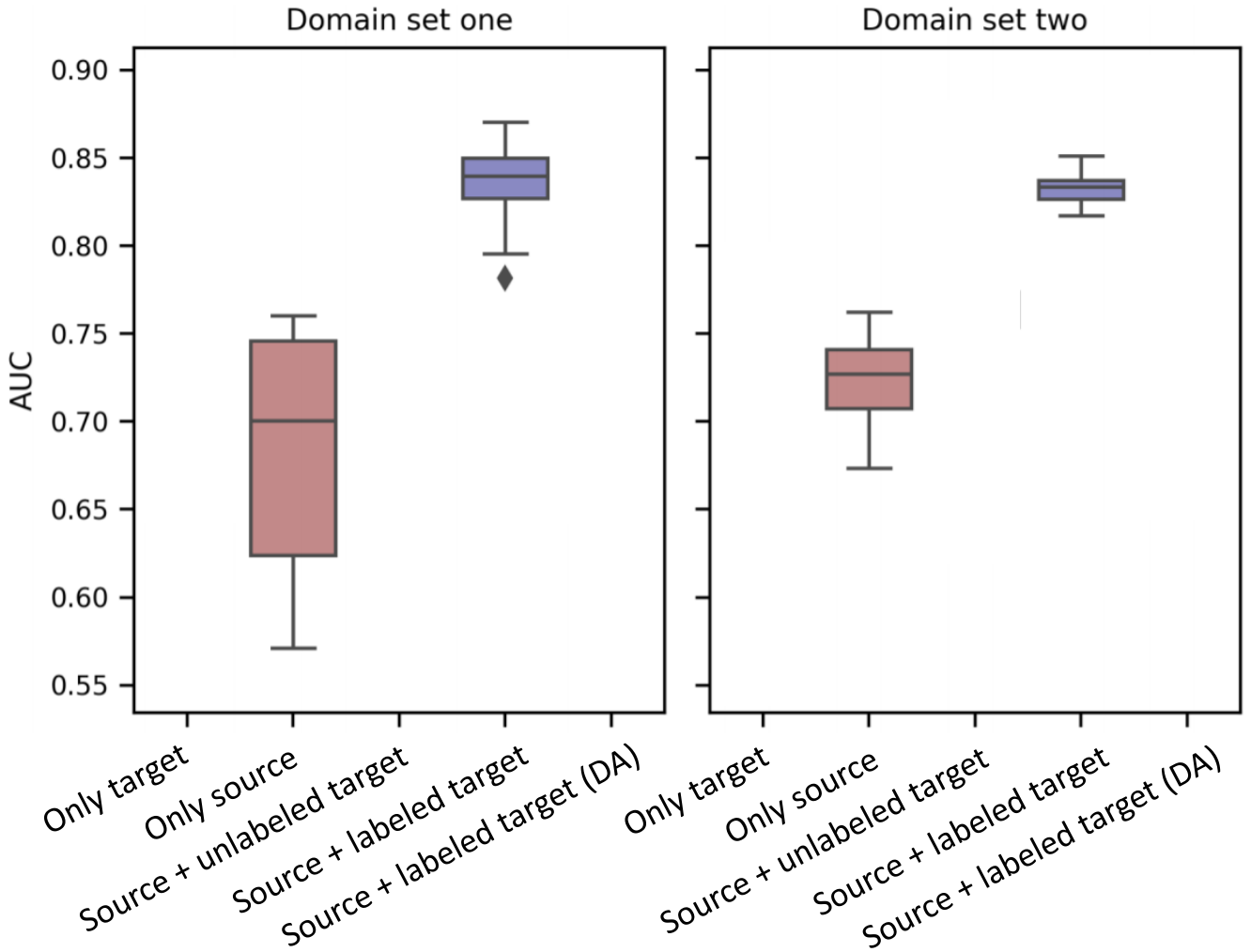


DA

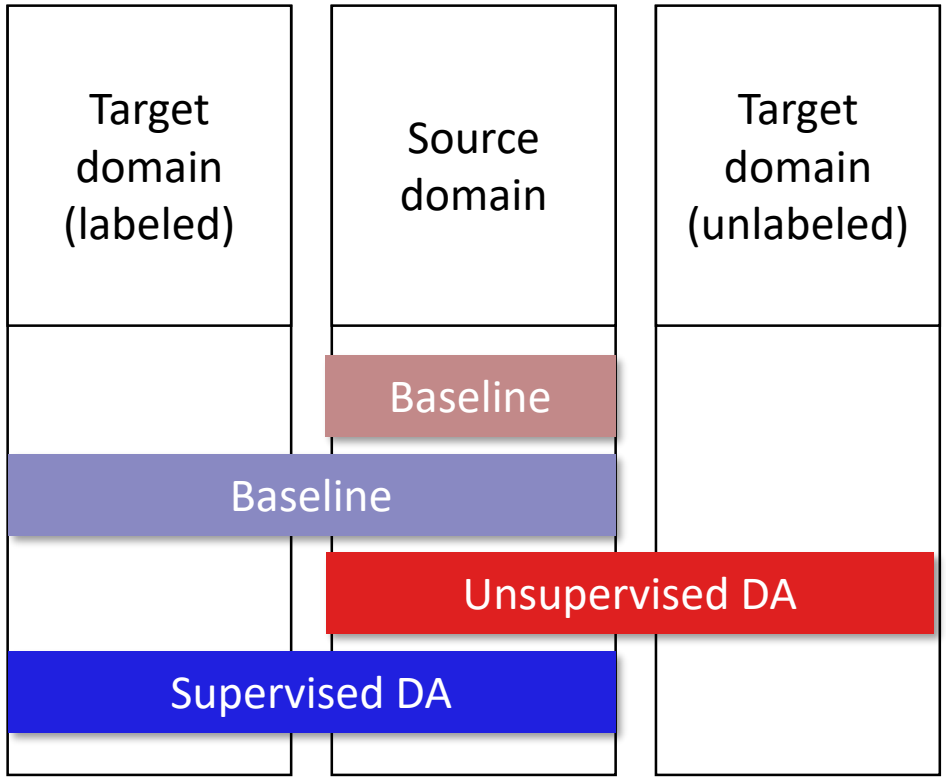


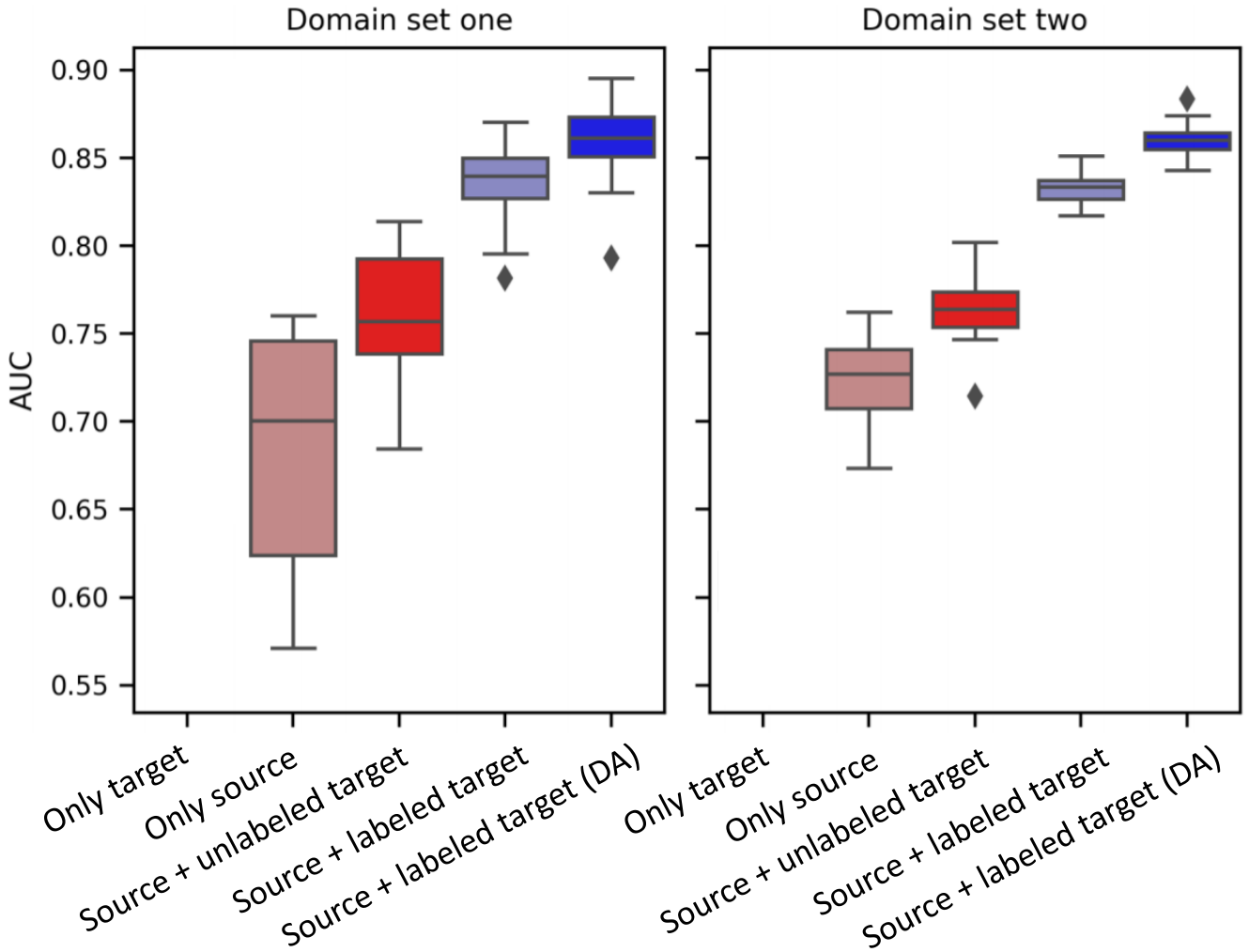
Generalize across domains for our Mighty Thymio





Generalize across domains for our Mighty Thymio

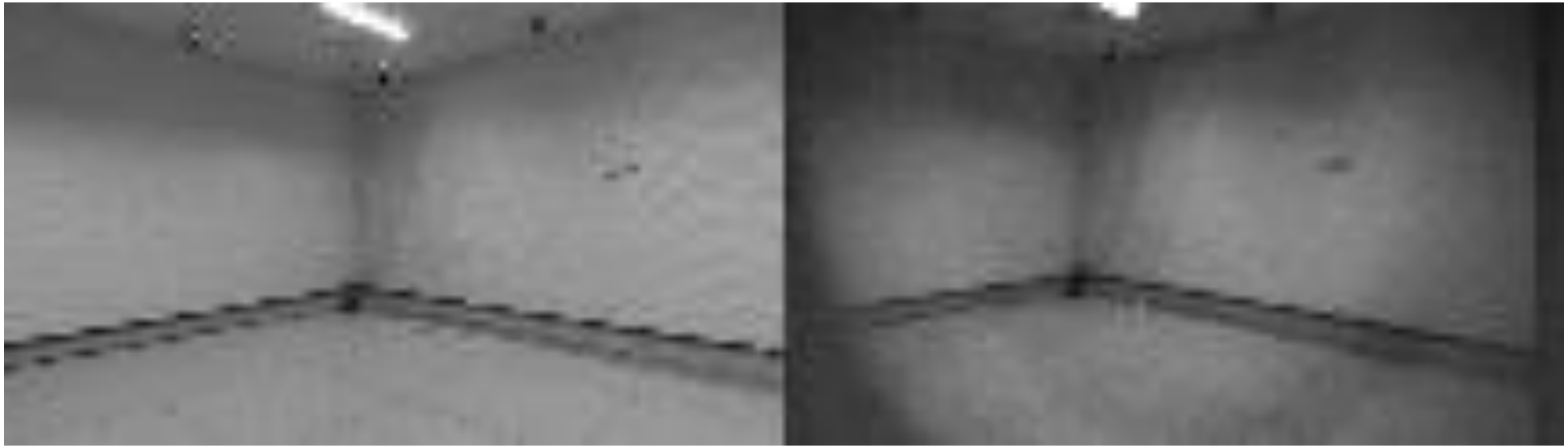




Domain Adaptation for proximity interaction

Source domain

Target domain



DA

Domain Adaptation for proximity interaction

Domain Adaptation: conclusions

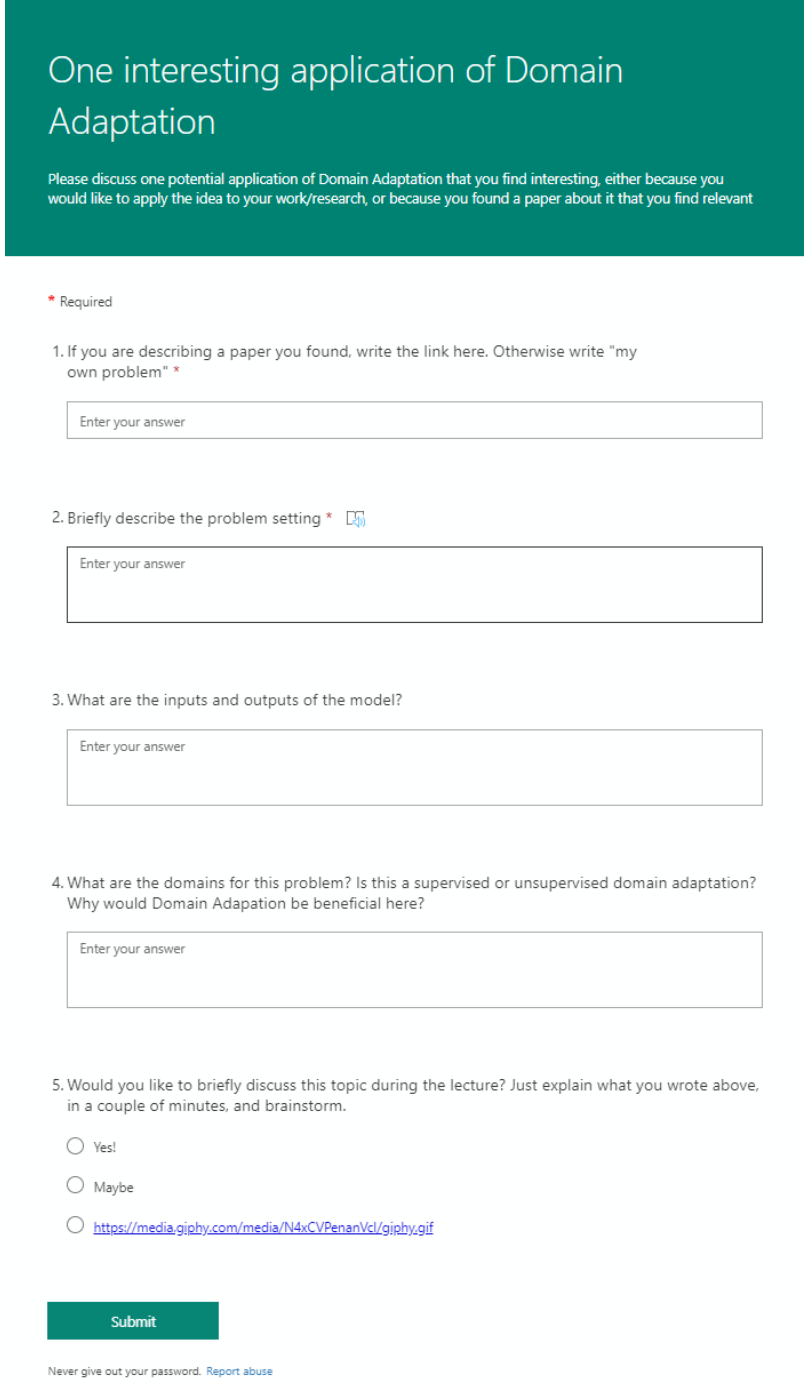
- Sooner or later, domain shift will screw your plans
- Domain adaptation to the rescue!
  - Semisupervised DA
  - Unsupervised DA
- “Frustratingly simple” methods, standard or deep
- Main ideas:
  - Align features across domains
  - Maximize domain confusion
  - Explicitly model domain shift

Now…

- Take another coffee!
- Fill another form!

https://forms.office.com/Pages/ResponsePage.aspx?id=pDglg56TEk-VLuM8-eVonUcsmPZBUUVBgjoOf_oZabpUN1ZHR0xKNU5SVVhIQ0ZUM1ZLUlBYNzVBVS4u

- See you at _____ for discussion



Further discussion and practical implications

https://www.nature.com/articles/s42256-020-0185-2

Read this and discuss