Multi-task, Multi-domain Learning: application to semantic segmentation and pose regression

Damien Fourure, Rémi Emonet, Elisa Fromont, Damien Muselet, Natalia Neverova, Alain Trémeau, Christian Wolf

Multi-task, Multi-domain Learning: application to semantic segmentation and pose regression

Damien Fourure[a,b,*], Rémi Emonet[a], Elisa Fromont[a], Damien Muselet[a], Natalia Neverova[b], Alain Trémeau[a], Christian Wolf[b]

[a] *Univ Lyon, LHC*
[b] *Univ Lyon, INSA-Lyon, LIRIS, F-69621, LYON, France*

Abstract

We present an approach that leverages multiple datasets annotated for different tasks (e.g., classification with different labelsets) to improve the predictive accuracy on each individual dataset. Domain adaptation techniques can correct dataset bias but they are not applicable when the tasks differ, and they need to be complemented to handle multi-task settings. We propose a new *selective loss* function that can be integrated into deep neural networks to exploit training data coming from multiple datasets annotated for related but possibly different label sets. We show that the gradient-reversal approach for domain adaptation can be used in this setup to additionally handle domain shifts. We also propose an auto-context approach that further captures existing correlations across tasks. Thorough experiments on two types of applications (semantic segmentation and hand pose estimation) show the relevance of our approach in different contexts.

Keywords: Deep learning, Convolutional neural networks, Semantic segmentation, Domain adaptation, Multi-task learning

*Corresponding author
Email addresses: damien.fourure@univ-st-etienne.fr (Damien Fourure), remi.emonet@univ-st-etienne.fr (Rémi Emonet), elisa.fromont@univ-st-etienne.fr (Elisa Fromont), damien.muselet@univ-st-etienne.fr (Damien Muselet), natalia.neverova@liris.cnrs.fr (Natalia Neverova), alain.tremeau@univ-st-etienne.fr (Alain Trémeau), christian.wolf@liris.cnrs.fr (Christian Wolf)

1. Introduction

Semantic scene parsing (a.k.a. semantic full scene labeling) from RGB images aims at segmenting an image into semantically meaningful regions, i.e., to provide a semantic class label for each pixel of an image — see Fig. 3 for examples of labels in an outdoor context. Semantic scene parsing is useful for a wide range of applications, for instance autonomous vehicles, automatic understanding and indexing of video databases, etc.

Most semantic scene parsing methods use supervised machine learning algorithms and thus rely on densely labeled (manually annotated) training sets which are very tedious to obtain. Only a small amount of training data is currently available for this task [1], which makes this problem stand out from other problems in vision (as for instance object recognition and localization). This is a particularly stringent problem for deep networks models which are particularly needy in terms of training data even if they have demonstrated their superior effectiveness to tackle this application [4]. In the case of depth images, data-augmentation using artificially-created training data has been employed successfully for segmentation problems [5, 6, 7]. However, the high variations of content in fully textured images make this solution at the moment very difficult to use for RGB images.

Most datasets for scene parsing contain only several hundreds of images, some of them only several dozen [8, 9, 10, 11, 12, 13, 14, 15, 16]. Additionally, combining these datasets is a non-trivial task as target classes are often tailored to a custom application. For example, one might be interested in specific types of vegetation like *trees*, *bushes* and *grass*, or types of objects such as *graffiti* or *billboard* while other applications do not require discriminating between these types. One good example of such labelset diversity can be found within the KITTI Vision benchmark [17, 18]. This dataset contains outdoor scene videos

[1] Since the CVPR 2016 conference, new datasets with larger amounts of data have been released ([1, 2, 3]), but we are still not near ImageNet level amounts of data.

acquired on roads around the city of Karlsruhe, in rural areas and on highways. Many works have been done on this dataset since its release in 2013, tackling computer vision tasks such as visual odometry, 3D object detection and 3D tracking [10, 11, 12, 13, 14, 15, 16]. To tackle these tasks, several research groups have labeled parts of the original dataset, independently from the other teams and often for different goals (among the works listed above, semantic segmentation is the final goal only for [12] and [16]). In practice, the ground truth segmentation quality varies and both the granularity and the semantics of the labels differ, even when some shared names (e.g., *vegetation*) are used as labels. However, all the existing labels can be useful to tackle the scene labeling problem. This inconsistency in the labelset is also true when using the Stanford Background [8] and SIFTFlow [9] datasets in combination with or in addition to KITTI.

The contributions of this paper are multiple: I) we formalize a simple yet effective *selective loss* function that can be used in deep networks to exploit training target data coming from multiple datasets with possibly different tasks (e.g., different labelsets). II) we show that the gradient-reversal approach for domain adaptation [19] can be used in this setup but needs to be manipulated with care especially when datasets are unbalanced, III) we propose an auto-context approach that further captures existing strong correlations across tasks, IV) we run thorough experiments on two types of applications (classification and regression) and on a total of 11 heterogeneously annotated datasets, underlining the impact of each part of the approach.

2. Related Works

In this section, we first discuss state-of-the-art methods dealing with multiple datasets, focusing in particular on feature and knowledge transfer methods in the context of deep networks. We then discuss semantic scene parsing with an emphasis on methods used for the KITTI benchmark.

Learning across datasets. Many recent papers [19, 20, 21, 22, 23] proposed methods to solve the problem of the transferability of deep features. Since CNNs require a lot of labeled data to provide very good features, the trend consists in exploiting features learned on one big dataset and in adapting them to other datasets and other tasks [22]. In an extensive analysis about deep feature transfer, Yosinski et al. [22] show that it is better to initialize lower layers from features learned on a different (and maybe distant) task than using random weights. These transferred features improve generalization performance even after fine-tuning on a new task. Hinton et al. [21] propose another way to transfer (or distill) knowledge from one large network to a smaller one. The idea is for the small network to learn both the outputs (soft targets) of the large network as well as the correct labels of the data. Accounting for the soft labels from the other network helps in learning the correlation between the labels. Furthermore, the authors showed that this distillation works even when the transfer set that is used to train the final small model, lacks samples of one or more of the classes.

Considering situations where the task is the same, but datasets are different(different distributions of the input data), the theory of domain adaptation tells us that the most similar the feature representation is across domains, the better the adaptation will be. Ganin and Lempitsky [19] follow this principle by learning features that are invariant with respect to the shift between the source and target domains. In order to do that, they train an additional *domain classifier* to discriminate the two domains from intermediate features. The goal is to train the parameters of the domain classifier to minimize its error, and to train the parameters of the feature representation to *maximize* the same error. This is in practise achieved by reversing the gradient during the backpropagation step.

The method proposed by Tzeng et al. [20] could seem very well suited to our problem. It merges the two previous ideas (soft labels and domain confusion) into a framework that allows to transfer network knowledge across domains and tasks and thus across datasets. However, the tasks proposed in the source and

target domain share the same labelset which makes the transfer of empirical category correlations from the source to the target significantly easier than in our case (where different label sets make distillation impossible). Still in the context of domain adaptation, Zhang et al. [23] propose to match the source and target marginal distributions of features as well as the source and target conditional distributions of the labels associated to the features. Therefore, while learning the target network, they minimize both the marginal and conditional empirical Maximum Mean Discrepancy (MMD) between the source and target distributions. The domain adaptation problem is a bit different from our current problem, where we have datasets labeled by different authors, with different and inconsistent labelings. Nevertheless, we can report three important points from the papers listed above: i) exploiting additional data improves generalization; ii) fine-tuning for each specific task improves the classification and iii) exploiting the correlations between the labels also helps the classification. These observations guide the proposed approach.

The method proposed in [24] to take into account different domains might seem similar to ours at first sight. However, they do not want to leverage multiple datasets (here video sequences) to improve single separated tasks across different domains but they train their network on a single subtask (here the classification of patches of a video as "target" vs "background") using multiple "datasets" (each one being a single video). Their method ultimately aims at providing good features to an online tracking algorithm. Their proposed architecture differs from ours (after the CNN of the joint training network shown in Figure 1b since it does not focus in identifying and benefiting from the correlations between the different tasks.

In [25], the authors propose a unified neural network-based framework to tackle at the same time the multi-domain and multi-task learning problems. They consider that meta data, called semantic descriptors, providing e.g. the domain and the task indexes, could be available for each example such that there is no need for further distinctions in the network between different domains and different tasks. Their network has 2 "sides", one dedicated to learn

a new representation from the original input vector and one that learns a representation from the meta data. Both sides outputs are combined in a single output layer at the end. Their objective function minimizes the empirical risk for all domains/tasks. If their method encompasses some of the works published on either multi domain or multi task learning before 2014 (when the paper was submitted) and gives good results on a zero-shot learning problem, it does not give any intuitions or theoretically founded results about how the proposed method could perform domain adaptation and how it could explicitly capture correlations between the task other than by sharing parameters in the network.

In this paper, we show how domain adaptation techniques can be integrated in our proposed method to benefit from multiple heterogeneously labeled datasets (which leads to different classifications tasks) without compromising the classification accuracy on each dataset.

Semantic Segmentation. Whereas the methods used for low level segmentation are diverse, high level semantic segmentation is dominated by machine learning. Learning methods range from random forests, to Support Vector Machines and deep networks. In [26] for instance, a structured-output version of random forests is proposed, where each leaf node predicts labels for every single pixel of an input patch, and predictions are integrated over patches using voting. Deep neural networks have also been used in wide range of works [27, 28, 29, 30].

Over the years, segmentation algorithms (semantic or not) have often been regularized through probabilistic graphical models like Markov Random Fields or Conditional Random Fields (CRF). Inference in these models requires to solve combinatorial problems which are often non-submodular and intractable. These methods have also been combined with machine learning, in particular deep networks [27, 31]. Regrouping pixels into larger structures, like superpixels, is also a frequently used technique [32, 27].

Auto-context models [33] are a different way to include structural information. They are defined as stacks of predictors, each one improving on the result of the previous, and have also been adapted for scene parsing. In [28], similarly

to recurrent networks for sequence classification, the same network is applied several times to outputs of different stages. In [29], a network is trained to predict context for a subsequent refinement network. Both networks are trained on segmentation maps, but the refinement network learns to cope with noisy segmentations as input. In [34], scene parsing from depth images using auto-context is formulated as a graphical model and solved through message-passing.

For the specific case of the KITTI dataset, [12] shows how to jointly classify pixels and predict their depth. The depth classifier only predicts the likelihood of a pixel to be at any canonical depth (binary problem) and the joint classifier is based on the multi-class boosted classifier suggested in [35]. In [16] the authors use a random forest (RF) classifier to classify segments of an image for different scales and sets of features (including depth information). Next, they train another RF classifier on the segments with overlapping coverage to fuse the unimodal classification results. Lastly they apply a CRF on the obtained results to enforce spacial consistency. None of the 7 cited methods used deep learning to tackle the semantic segmentation step. The aim of this paper is to show how to learn across datasets (with possibly different tasks) to improve the classification results. In particular, we do not optimize our method to produce the best accuracy for each of the used dataset. For example, while in KITTI many authors use rich features such as color, depth and temporal features, and complex post processing, we only use the RGB channels for our experiments and we do not take the label hierarchies into account. We also do not use any post-processing for the Stanford Background and Sift-flow datasets. For the sake of completeness we will still provide the state-of-the-art results for these datasets.

Recent methods explored a weakly supervised setting to alleviate the problem of manual annotations [36, 37]. Instead of requiring pixel-wise ground-truth, they integrate image-wise information, or point-wise ground-truth which can be easily provided. They are usually strongly regularized through priors like objectness [36] or classification performance based on the full image [37].

This paper extends [38]. We introduced another method (called *Joint train-*

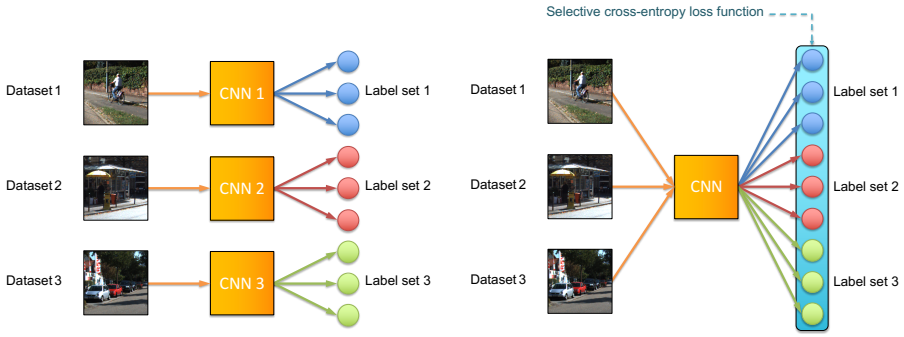
ing with shared context) that can use the correlations between the labelsets learned by the network to improve the accuracy. We showed with much more experiments that this strategy improves the results when the labelsets are correlated. Finally, we have also tested our method on another totally different application: a hand pose estimation problem. With this new application we have shown that our method can be successfully applied to both classification and regression problems.

3. Proposed Approach

Problem statement: multi-task multi-domain learning. Given a set of images drawn from a set of K different datasets, pairs made of an input patch x_i^k and a target label y_i^k are grouped into sets $D_k = \{x_i^k, y_i^k\}$, where $k{=}1 \dots K$ and i indexes patches. The label spaces are different over the datasets, therefore each y_i^k can take values in space $\mathcal{L}^k$.

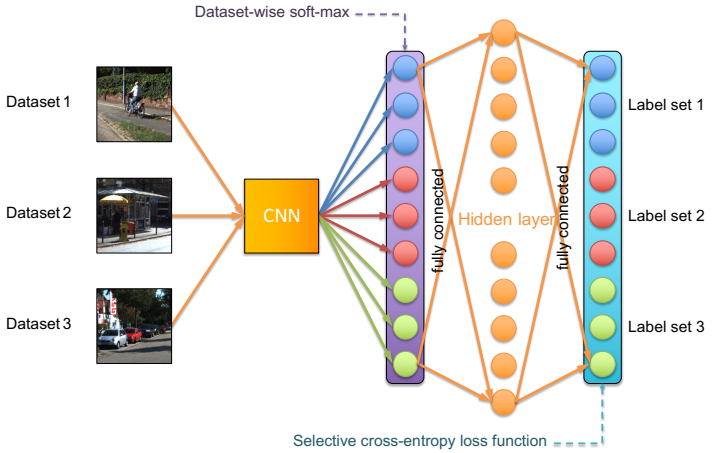
The problem is a multi-task problem: the tasks may be of the same nature (e.g., classification) but they are genuinely different tasks. The problem differs from a multi-label classification: instead of predicting the presence or absence of each label, we must predict one and exactly one label from each labelset $\mathcal{L}^k$. Merging datasets also yields a domain adaptation problem: all datasets may be of the same kind (e.g., RGB images) but the input distributions vary vastly across datasets.

Our goal is to learn a nonlinear mapping $\hat{y} = \theta(x, \Theta)$ with parameters Θ which minimizes a chosen risk $\mathcal{R}(y, \hat{y})$. The mapping θ is represented as a convolutional neural network, where each layer itself is a nonlinear mapping $f_l(\mathbf{W}_l \mathbf{h}_{l-1} + \mathbf{b}_l)$ where $\mathbf{h}_l$ is the l^{th} hidden representation, $\mathbf{W}_l$ and $\mathbf{b}_l$ are the weights and bias of the l^{th} layer and $f_l(.)$ is the activation function of the l^{th} layer. We minimize the empirical risk, $\mathcal{R}(\theta(x, \Theta), y) = \frac{1}{N} \sum_{k=1}^{n} J(x, y, \Theta)$, where N is the number of training samples and J is the loss function for each individual sample. We use the cross entropy loss $J(x, y, \Theta) = -\sum_j 1_{y=j} \log \theta(x, \Theta)_j$, where $\theta(x, \Theta)_j$ is the network output for class j.



(a) No fusion

(b) Joint training



(c) Joint training with shared context

Figure 1: 1a,1b and 1c show our different strategies. 1a, named *No Fusion* is our baseline and consist of learning one network per dataset. 1b, named *Joint training*, consist of learning only one network with our selective loss function. 1c named *Joint training with shared context* add a Multi-Layer Perceptron after the network 1b (already learned) and fine tune it using all the datasets.

Limitations of separate training. Considering K different datasets, the classical baseline approach is to train K separate mappings (models) θ^k, each defined on its own label set $\mathcal{L}^k$. This baseline approach is illustrated in Figure 1a. Unfortunately this basic approach presents several shortcomings:

(i) Each mapping θ^k is trained on its own dataset D^k, which requires minimizing over a separate sets of parameters Θ^k. In the chosen deep convolutional implementation the parameters $\Theta^k=\{\mathbf{W}^l, \mathbf{b}^l\}_{l=1}^L$ include all convolution's filters and the weights and bias of all fully connected layers, which are generally large sets (more than 2 millions parameters). Learning such a large amount of parameters from limited (and generally small) amounts of training data is very challenging.

(ii) Relationships between label spaces are neither modeled nor exploited, which further limits the power of the trained models.

Joint feature training with selective loss. We propose to tackle shortcoming (i) by exploiting the hierarchical nature of deep models. It is well known that, on most classical problems in computer vision, supervised training leads to a rising complexity and specificity of features over layers [39]. In our case, we propose to train a single deep network on the union of all individual datasets. This allows the network to decide at every layer which features should be generic and which ones should be task-specific. Note, that we do not force this separation into private and shared subspaces explicitly. However, given that we training proceeds by sampling from different subsets, features which are useful for multiple subsets will arise naturally. Given sufficient network capacity, they will be completed by features which allow to minimize loss on individual subsets.

This joint training approach is illustrated in Figure 1b. There is one output unit per label in the union of all label sets $\mathcal{L}^k$, which means that the output layer is able to provide predictions for any of the considered datasets.

In a traditional multi-class setting, the network output is computed using a soft-max function (as the activation function of the last layer) to produce a

probability vector. This is used to stochastically optimize the likelihood of the target labels. However, with K different datasets, this is counter-productive: it maximizes the target class probability but minimizes the probability of all other classes, including the ones from different label sets. This minimization is problematic when there exists a correlation between labels across different datasets. As a concrete example, in the KITTI dataset (see Fig. 3 where all labels are reported) the class *Tree* of the dataset from He et al. [10] is likely correlated with the class *Vegetation* from the dataset labeled by Kundu et al [11]. A plain softmax, optimizing the probability of the *Tree* class will implicitly penalize the probability of *Vegetation*, which is not the desired effect.

We thus define the *dataset-wise soft-max* (that produces a probabilities vector per dataset): for each label j from dataset k,

$$f(j, \theta(x, \Theta)) = \frac{e^{\theta(x,\Theta)_j}}{\sum_{j' \in \mathcal{L}^k} e^{\theta(x,\Theta)_{j'}}} \tag{1}$$

In practice, during learning, the dataset-wise soft-max is combined with a cross-entropy loss function to build what we call the *Selective cross-entropy* loss function :

$$J'(k, x, y, \Theta) = -\theta(x, \Theta)_y + \log(\sum_{j \in \mathcal{L}^k} e^{\theta(x,\Theta)_j}) \tag{2}$$

Cross-entropy is a standard loss for classification tasks, which maximizes the negative log likelihood of the winning class. In practice, and for efficiency reasons, both the standard cross entropy and the dataset-wise soft-max are combined (Eq. 2) into a single computational layer. During the training phase, the network parameters are updated classically using SGD and gradient backpropagation. From Equation 2 it can be seen that gradients are zero for parameters involving output units corresponding to labels from datasets l with $l \neq k$ meaning that they are not penalized by the weight updates. This is equivalent to using separate output layers for each dataset and intermediate layers with shared parameters over the different datasets.

Modeling correlations between label-sets. Shortcoming (ii) is partly addressed by the joint feature training method, as correlations between labels across datasets can be learned by the shared layers in the network. On the other hand, we will show that explicitly modeling these correlations further improves the discriminative power of the classifier.

To take into account the correlation between labelsets, we add an additional fully-connected layer to the network $\theta(x, \Theta)$ learned with our *Joint training* strategy. The new mapping $\theta'(x, \Theta)$ is trained using the same selective loss function than for the *Joint training* strategy. This *Joint training with shared context* approach is illustrated in Figure 1c. Note, that in this setting dataset selection is performed twice: selective loss is used at the output layer of the new network, and the dataset-wise soft-max activation function is also used at the layer corresponding to the output of the pre-trained network before additional fully connected layers were added.

Gradient Reversal for Domain Adaptation. So far, our method trains a single network on several datasets adapting for different labelsets, while ignoring eventual shifts in the input domain between datasets. This is not a problem in the case where the input data is sampled from a single distribution (e.g., for the different subsets of the KITTI dataset). In other cases, a non neglectable shift in input distribution does exist, for instance between the Stanford and SIFTFlow data.

The theory of domain adaptation tells us that a better adaptation between source and target domains can be achieved when the feature distributions of both sets are closer to each other. In the lines of Ganin and Lempitsky [19], this can be achieved using an additional classifier trained on the same features, which attempts to predict the domain of the input data. In the case of domain invariant features, this classifier will achieve high error.

More precisely, our full mapping $y = \theta(x, \Theta)$ is conceptually split into two parts: the *feature extractor* $f = \theta_f(x, \Theta_f)$, which corresponds to the first convolutional layers and results in features f, and the *task classifier* $y = \theta_t(f, \Theta_t)$,

which corresponds to the later fully connected layers including the selective loss layer. The *domain classifier* is an additional mapping, which maps the features f to an estimated domain $d = \theta_d(f, \Theta_d)$. The goal here is to *minimize* the loss of the domain classifier θ_d over its parameters Θ_d in order to train a meaningful classifier, and to *maximize* the same loss over the features f, i.e., over the parameters Θ_f of the feature extractor θ_f, in order to create domain invariant features. In the lines of [19] this is achieved through a single backpropagation pass over the full network implementing θ_f, θ_t and θ_d, inversing the gradients between the domain classifier θ_d and the feature extractor θ_f. In practice, the gradient is multiplied with a hyper-parameter $-\lambda$, which inverse the gradient and controls the importance of the task classifier and the domain one.

Our experiments described in Section 4 show, that this domain adaptation step is also useful and important in our more general setting where results are requested for different labelsets.

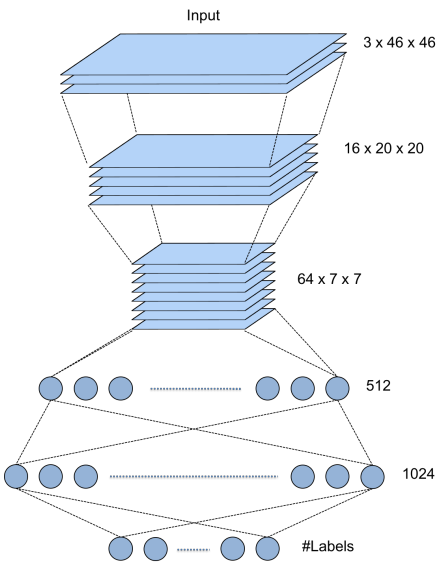
4. Experimental Results

Our experiments use the Torch7 [40] framework and training was operated on NVIDIA Titan-X GPUs.

4.1. Training details for Semantic Segmentation

For all semantic segmentation experiments we used a network architecture inspired by Farabet et al.[27] for outdoor scene labeling and improved with the recent advances in deep neural network research. Our network is composed by 3 convolutional layers followed by 2 fully-connected layers.

The network is illustrated in Figure 2. The first two convolutional layers are composed by a bank of filters of size 7×7 followed by ReLU [41] units, 2×2 maximum pooling and batch normalization [42] units. The last convolutional layer is a filter bank followed by a ReLU unit, a batch normalization unit and dropout [43] with a drop factor of 30%. The first fully connected linear layer is then followed by a ReLU unit and the last layer is followed by our dataset-wise



name	kernel size	output size
input	-	$3 \times 46 \times 46$
batch-norm-0	-	$3 \times 46 \times 46$
conv-1	7×7	$16 \times 40 \times 40$
relu-1	-	$16 \times 40 \times 40$
max-pooling-1	2×2	$16 \times 20 \times 20$
batch-norm-1	-	$16 \times 20 \times 20$
conv-2	7×7	$64 \times 14 \times 14$
relu-2	-	$64 \times 14 \times 14$
max-pooling-2	2×2	$64 \times 7 \times 7$
batch-norm-2	-	$64 \times 7 \times 7$
conv-3	7×7	$512 \times 1 \times 1$
relu-3	-	$512 \times 1 \times 1$
batch-norm-3	-	$512 \times 1 \times 1$
dropout-4	0.3	512
fully-connected-4	-	1024
relu-4	-	1024
fully-connected (output)		$\#labels$

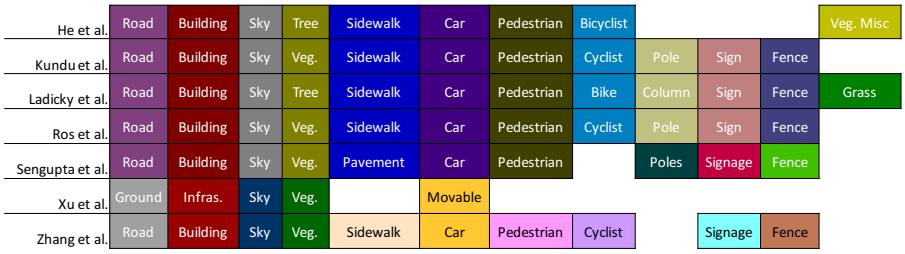
Figure 2: The CNN network (orange boxes in Figure 1) architecture used for semantic segmentation experiments

softmax unit. For the strategy illustrated in Figure 1c, an additional fully connected hidden layer is added followed by a ReLU unit and a batch normalization unit. Our dataset-wise softmax unit is also used for this output layer.

To train the network, RGB images are converted to the YUV color space. A training input example is composed of a patch x_i of size 46x46 cropped from an image, the dataset k from which the image comes from, and y_i^k, the label of the center pixel of the patch x_i. Stochastic gradient descent with a mini-batch of size 128 was used to update the parameters. We used early stopping on a validation set in order to stop the training step before over-fitting. The only data augmentation strategy that we used is a horizontal flip of patches.

4.2. Datasets details

The KITTI dataset has been partially labeled by seven research groups resulting in 736 labeled images that are split into a train set, a validation set and a test set. When the information was given by the author, we used the same train/test set as them, otherwise we randomly split them into approximately



Data	Train	Val	Test	Total
He [10]	32	7	12	51
Ku [11]	28	7	15	50
La [12]	24	6	30	60
Ro [13]	80	20	46	146
Se [14]	36	9	25	70
Xu [15]	56	14	37	107
Zh [16]	112	28	112	252
Total	368	91	277	736

Figure 3: The 68 labels (with the original colors) used by the different authors to annotate their subset of the KITTI benchmark as well as the number of images (and their train/validation/split decomposition, see details in Section 4.2) in each subset.

70% of data for the training and validation sets and 30% data for the test set, ensuring that any two images from the same video sequence end up in the same split. The labels used in the different subsets of the KITTI dataset are summarized in Fig. 3. Note that Xu et al. [15] provide a complete hierarchy of very detailed labels, but we only used the highest level of the hierarchy to obtain a labeling more compatible (in terms of granularity) with the labels from the other research teams. The KITTI dataset contains over 40000 frames (180GB of raw videos) but in this work we only rely on the labeled data. We then sample on average 390000 patches in each video frame (depending on its size). This results into a dataset of about 280 million patches suitable to train a deep learning architecture. As mentioned in Section 2, the different labels provided by the different teams are not always consistent. As illustration in Fig. 3, we can see that the granularity and the semantics of the labels may be very different from one labeling to another. For example, Ladicky et al. separate the *Trees* from the *Grass*. However, this might correspond to the *Vegetation* labels

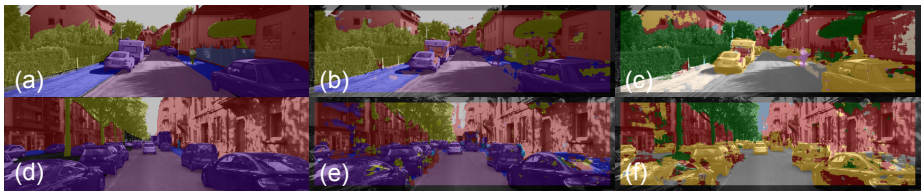


Figure 4: Example of pixel classification results given by the JTSC strategy (see Figure 1). **(a)** is the ground truth from the Ros et al. labelset. **(b)** is the result obtained with our strategy for the same labelset and **(c)** is the result obtained for the Kundu et al. labelset. **(d)** is a ground truth image from He et al.. **(e)** is the result obtained with our strategy for the same labelset and **(f)** is the result obtained for the Xu et al. labelset.

in the subset from Xu et al. but might also correspond (in the case of *Grass*) to the labels *Ground*. He et al. [10] have not used the labels *Pole*, *Sign* or *Fence* used in most other labelings. These labels are likely to overlap with the label *Building* of He et al. but then, this *Building* class cannot be consistent anymore with the other labelings that contain the label *Building* in other subsets. Some groups have used the label *Bike* and some others have used the label *Cyclist*. Those two labels are likely to overlap but in one case a team has focused on the entire entity "cyclist on a bike" whereas another has only focused on the bike device.

In addition to the KITTI dataset, our approach has been tested on two scene labeling datasets: Stanford Background [8] and SIFTFlow [9]. The Stanford Background dataset contains 715 images of outdoor scenes having 9 classes. Each image has a resolution of 320×240 pixels. We randomly split the images to keep 80% of them for training and 20% for testing. From these images, we extract a total of 40 millions patches. The SIFTFlow dataset contains 2688 manually labeled images of 256×256. The dataset has 2488 training and 200 test images containing 33 classes of objects. From this we extract 160 millions patches.

	7 KITTI		
Methods	Global	Avg.	IoU
NoFusion	77.35	54.51	41.99
JT	80.71	58.62	46.21
JT + DE	80.84	58.89	46.32
JTSC	**81.75**	**59.67**	**47.10**

	Learning with Stanford + SIFTFlow								
	Stanford			SIFTFlow			Accuracy on both		
Methods	Glob.	Avg.	IoU	Glob.	Avg.	IoU	Glob.	Avg.	IoU
No Fusion	74.80	64.35	51.41	69.79	24.95	19.02	72.66	33.24	25.84
JT	75.79	67.22	**54.43**	71.55	28.34	21.35	73.99	36.52	28.31
JT+GR	75.07	67.76	53.80	70.82	25.08	19.06	73.26	34.07	26.37
JT+DE	75.28	67.37	52.93	71.11	**29.01**	**22.25**	73.51	**37.08**	**28.71**
JT+GR+DE	**76.04**	67.71	54.05	**71.65**	27.40	20.54	**74.17**	35.89	27.60
JTSC+GR+DE	75.45	**68.33**	53.83	71.39	28.03	21.09	73.73	36.51	27.98

	Learning with 7 KITTI + Stanford								
	7 KITTI			Stanford			Accuracy on both		
Methods	Glob.	Avg.	IoU	Glob.	Avg.	IoU	Glob.	Avg.	IoU
No fusion	77.35	54.51	41.99	**74.80**	64.35	51.41	77.09	55.64	43.10
JT	80.40	58.42	45.93	73.38	63.42	50.51	79.64	58.97	46.43
JT+GR	80.73	58.33	46.08	72.42	62.79	49.31	79.82	58.82	46.43
JT+DE	80.76	58.65	46.24	73.93	63.87	51.19	80.02	59.23	46.78
JT+GR+DE	81.20	58.25	46.28	74.39	**65.31**	**52.04**	80.46	59.02	46.91
JTSC+GR+DE	**81.26**	**60.24**	**47.92**	74.70	63.44	50.88	**81.08**	**60.59**	**48.24**

	Learning with 7 KITTI + SIFTFlow								
	7 KITTI			SIFTFlow			Accuracy on both		
Methods	Glob.	Avg.	IoU	Glob.	Avg.	IoU	Glob.	Avg.	IoU
No fusion	77.35	54.51	41.99	69.79	24.95	19.02	76.72	45.18	34.73
JT	79.86	57.05	44.94	70.67	**28.65**	**21.44**	79.1	48.08	37.52
JT+GR	80.47	57.10	45.31	70.60	28.56	21.39	79.65	48.08	**37.76**
JT+DE	78.93	53.94	42.63	**72.15**	27.24	21.12	78.37	45.51	35.84
JT+GR+DE	80.44	**58.12**	**45.56**	70.88	27.11	20.4	79.65	**48.33**	37.62
JTSC+GR+DE	**82.07**	54.68	44.87	70.89	23.11	17.76	**81.15**	44.71	36.31

Table 1: Pixel (*Global*) and Class (*Average*) accuracy and Intersection over Union (IoU) results for the 7 used subsets of the KITTI dataset (7 KITTI) and combinations of each pair of datasets: SIFTFlow + Stanford; 7 KITTI + Stanford; 7 KITTI + SIFTFlow with different training strategies: NF=No Fusion (see Fig. 1a) ; JT= Joint training (see Fig. 1b); DE=Dataset Equilibrium; GR=Gradient Reversal; JTSC=Joint training with shared context (see Fig. 1c). Best results are highlighted in bold. For clarity we give only the global results of the 7 sub-datasets of KITTI. Tables 2,3 and 4 give the detailed results for respectively 7 KITTI (blue table), 7 KITTI + Stanford (orange table) and 7 KITTI + SIFTFlow (green table)

Methods		7 KITTI							
		He [10]	Kundu [11]	Ladicky [12]	Ros [13]	Sengupta [14]	Xu [15]	Zhang [16]	Total
No Fusion	Global	76.11	71.36	73.48	76.80	73.25	86.07	77.30	77.35
	Average	59.66	53.58	42.81	48.46	64.25	81.96	49.53	54.51
	IoU	49.65	39.80	31.74	38.60	44.32	70.83	37.53	41.99
Joint Training	Global	79.46	76.91	75.89	78.22	78.65	87.94	81.85	80.71
	Average	64.42	60.01	**45.34**	50.23	71.70	84.37	53.10	58.62
	IoU	54.35	**46.01**	**34.08**	40.69	51.27	74.31	41.93	46.21
Joint Traning with Dataset Equilibrium	Global	**80.11**	**77.91**	**76.53**	77.43	80.42	87.84	81.77	80.84
	Average	64.33	**59.66**	43.87	50.93	74.14	84.76	**53.90**	58.89
	IoU	55.24	45.65	33.95	39.37	53.24	74.19	42.14	46.32
Joint Training with shared context	Global	78.56	77.34	75.56	**78.53**	**81.34**	**88.19**	**84.04**	**81.75**
	Average	**64.99**	59.37	44.92	**52.96**	**75.69**	**84.80**	53.79	**59.67**
	IoU	**57.83**	45.81	34.04	**40.98**	**54.83**	**74.62**	**43.86**	**47.10**

Table 2: Detailed accuracy results for different learning strategies on 7 subsets of the KITTI datasets. Best results are highlighted in bold.

Methods		Learning with 7 KITTI + Stanford									
		He [10]	Kundu [11]	Ladicky [12]	Ros [13]	Sengupta [14]	Xu [15]	Zhang [16]	7 KITTI	Stanford [8]	Total
No Fusion	Global	76.11	71.36	73.48	76.80	73.25	86.07	77.30	77.35	**74.80**	77.09
	Average	59.66	53.58	42.81	48.46	64.25	81.96	49.53	54.51	64.35	55.64
	IoU	49.65	39.80	31.74	38.60	44.32	70.83	37.53	41.99	51.41	43.10
Joint Traning	Global	78.80	77.00	75.92	**78.45**	78.32	88.26	80.83	80.40	73.38	79.64
	Average	63.35	60.38	44.44	50.60	71.37	84.81	53.07	58.42	63.42	58.97
	IoU	53.44	46.38	33.65	41.07	50.37	74.90	41.10	45.93	50.51	46.43
Joint Training with Gradient Reversal	Global	78.89	77.16	75.62	78.33	79.39	87.99	81.76	80.73	72.42	79.82
	Average	62.66	**60.74**	44.78	**51.16**	71.44	84.90	51.49	58.33	62.79	58.82
	IoU	53.31	**46.66**	33.59	**41.35**	51.37	74.59	40.90	46.08	49.31	46.43
Joint Training with Datasets Equilibrium	Global	80.79	77.77	77.02	77.00	81.45	87.74	81.33	80.76	73.93	80.02
	Average	64.88	60.01	44.68	49.40	74.79	84.72	51.72	58.65	63.87	59.23
	IoU	55.49	45.57	34.42	38.82	54.39	73.85	40.69	46.24	51.19	46.78
Joint Training with Datasets Equilibrium and Gradient reversal	Global	81.41	77.34	77.52	77.02	79.50	88.26	82.65	81.20	74.39	80.46
	Average	64.56	59.01	44.60	49.35	71.12	84.77	53.77	58.25	**65.31**	59.02
	IoU	55.48	44.95	34.60	39.77	51.67	74.79	42.30	46.28	**52.04**	46.91
Joint Traning with shared context	Global	**82.13**	**78.17**	**77.88**	77.93	**82.91**	**88.38**	**82.82**	**81.26**	74.70	**81.08**
	Average	**67.44**	60.02	**46.05**	50.57	**76.51**	**84.93**	**55.40**	**60.24**	63.44	**60.59**
	IoU	**58.09**	45.93	**35.65**	40.83	**56.19**	**74.98**	**43.31**	**47.92**	50.88	**48.24**

Table 3: Detailed accuracy results for different learning strategies on Stanford dataset and 7 subsets of the KITTI datasets. Best results are highlighted in bold.

Methods		Learning with 7 KITTI + SIFTFlow									
		He [10]	Kundu [11]	Ladicky [12]	Ros [13]	Sengupta [14]	Xu [15]	Zhang [16]	7 KITTI	SIFTFlow [9]	Total
No Fusion	Global	76.11	71.36	73.48	76.80	73.25	86.07	77.30	77.35	69.79	76.72
	Average	59.66	53.58	42.81	48.46	64.25	81.96	49.53	54.51	24.95	45.18
	IoU	49.65	39.80	31.74	38.60	44.32	70.83	37.53	41.99	19.02	34.73
Joint Traning	Global	80.39	76.97	74.81	76.73	76.92	87.28	81.18	79.86	70.67	79.10
	Average	**63.97**	61.18	43.47	48.27	66.96	84.05	50.89	57.05	**28.65**	48.08
	IoU	**54.31**	**46.43**	32.79	38.81	48.39	73.15	40.06	44.94	**21.44**	37.52
Joint Training with Gradient Reversal	Global	80.14	76.61	76.08	76.45	80.81	87.60	81.63	80.47	70.60	79.65
	Average	63.82	60.04	43.31	48.37	**70.40**	83.80	49.58	57.10	28.56	48.08
	IoU	54.02	45.44	33.08	38.41	**52.47**	73.54	39.90	45.31	21.39	**37.76**
Joint Training with Datasets Equilibrium	Global	79.60	74.84	74.26	74.24	75.71	86.68	81.05	78.93	**72.15**	78.37
	Average	63.02	55.85	41.88	41.88	61.08	81.71	48.93	53.94	27.24	45.51
	IoU	52.69	41.70	31.05	31.05	47.13	71.08	39.45	42.63	21.12	35.84
Joint Training with Datasets Equilibrium and Gradient reversal	Global	80.25	76.92	76.09	77.25	78.26	87.92	81.50	80.44	70.88	79.65
	Average	63.56	**61.54**	**44.82**	**50.05**	69.26	84.42	**51.96**	**58.12**	27.11	**48.33**
	IoU	53.61	46.25	**33.86**	**39.64**	49.87	74.05	40.89	**45.56**	20.40	37.62
Joint Traning with shared context	Global	**81.91**	**79.56**	**76.92**	**77.54**	**80.81**	**88.31**	**84.31**	**82.07**	70.89	**81.15**
	Average	63.28	54.98	42.11	45.15	61.59	**84.69**	51.84	54.68	23.11	44.71
	IoU	54.21	43.62	32.92	36.60	49.45	**74.78**	**42.98**	44.87	17.76	36.31

Table 4: Detailed accuracy results for different learning strategies on SIFTFlow dataset and 7 subsets of the KITTI datasets. Best results are highlighted in bold.

4.3. Segmentation results with different training strategies

Table 1 shows the results obtained for all our training strategies (Tables 2,3 and 4 give detailed results for the 7 sub-datasets of KITTI). We report the *global accuracy*, the *average accuracy* and the *Intersection over Union*(IoU) measures. *Global* is the number of correctly classified pixels (*True Positive*) over the total number of pixels (also called *recall* or *pixel accuracy*), *Average* is the average of this recall per class (also called the *class accuracy*) and IoU is the ratio $TP/(TP+FP+FN)$ where TP is the number of *True Positive* and FP, FN are respectively the *False Positive* and *False Negative* averaged across all classes (this is the Jaccard Index, commonly known as the PASCAL VOC Intersection-over-Union measure). Note that the last column (*accuracy on both*) gives the global, average and IoU accuracies for all datasets together so that the totals take into account the relative number of labeled pixels in each dataset instead of being the average of all elements in the corresponding row.

No Fusion. The first learning implemented strategy consists in learning one network per dataset with the architecture described in Section 4 and illustrated in Figure 1a. This is our baseline, and the results for this strategy are shown in the rows described as *No Fusion*. State-of-the-art performance for the different KITTI sub datasets are (respectively for global and average accuracies): $(92.77, 68.65)$ for He et al. [10]; (97.20, non reported) for Kundu et al. [11]; $(82.4, 72.2)$ for Ladicky et. al. [12]; $(51.2, 61.6)$ for Ros et al. [13]; $(90.9, 92.10)$ for Sengupta et al. [14]; (non reported, 61.6) for Xu et al. [15]; and $(89.3, 65.4)$ for Zhang et al. [16]. For Stanford with 8 classes (resp. SIFTFlow), [31] reports a global accuracy of 82.3 (resp. 80.9), a pixel accuracy of 79.1 (resp. 39.1) and an IoU of 64.5 (resp. 30.8). These results are better (except for Ros et al. in Table 1) than those reported in our tables. This can be explained by the fact that either [11, 14, 12, 13, 31] only show results computed from a subset of their labels (e.g., the label *pedestrian* is ignored in [14, 12, 13]) and/or the features used by all methods on KITTI are richer (e.g., depth and time) and/or the proposed methods always combine multiple classifiers tuned on one particular

sub-dataset. To be able to assess our contributions, we believe that the *No Fusion* baseline is the fairest baseline.

Joint Training (JT). The second alternative strategy (*Joint Training*) consists in learning one single network (illustrated in Figure 1b) with all the datasets using the selective loss function detailed in Section 3. We can see that this strategy gives better results than our baseline for all combination of datasets (for example, in Table 1, learning with all the subsets of KITTI gives, on average for all the 7 subsets, an improvement of +3.36 on the Global accuracy, of +4.11 on the Average accuracy and of +4.22 on the IoU). These results show that this strategy allows us to increase the number of data used to train the network even if the labelsets of the datasets are different. This cross-task data augmentation leads to a better generalization and so to better classification accuracies. The only exception lies when the Stanford dataset is trained together with the 7 subsets of KITTI (in this case, all the performance results drop of about 1 point when using the Joint Training approach). We believe that this means that the distribution shift from Stanford to KITTI is too big and the size of the datasets too unbalanced (KITTI $\gg$ Stanford) to really help classifying images from the Stanford dataset. However, for the sake of completeness and to evaluate the contribution of the selective loss over the mere augmentation of data, we trained our network with pairs of datasets where one dataset was used to initialize the weights of the convolutional part of the network and the other (usually the smaller one) was used to fine tune the network. The results reported in Table 5 consistently show that this fine-tuning approach increases all the performance measures but at a lower extent than when using our method.

Joint Training with Shared Context (JTSC). The possible correlations between labels are taken into account in the *Joint training with shared context* strategy, illustrated in Figure 1c and detailed in Section 3. This approach improves accuracy for most sub datasets of KITTI (see Tables 2,3 and 4) when the KITTI subsets are used alone but gives comparable results when the learning is done jointly with the other different datasets. This shows that the approach is es-

Evaluation (and fine tuning) on SIFTFlow				
		Global	Average	IoU
Pre-Trained on	Stanford	70.86	23.52	18.05
	7 KITTI	70.33	27.07	19.87
Our methods	No Fusion	69.79	24.95	19.02
	Best method	71.11	29.01	22.25

Evaluation (and fine tuning) on Stanford				
		Global	Average	IoU
Pre-Trained on	SIFTFlow	74.50	65.65	52.52
	7 KITTI	73.57	63.82	50.81
Our methods	No Fusion	74.80	64.35	51.41
	Best method	76.04	67.71	54.05

Table 5: Results obtained on the SIFTFlow (top table) and Standford (bottom table) datasets when we pretrained the network on one dataset and fine tune it on an other in comparison to training it directly using pairs of dataset with our selective loss function.

pecially relevant when the labels between datasets are highly correlated. We studied the confusion matrices for the KITTI dataset: they show that errors made by the first network were reduced by better (i.e., correct) predictions with the JTSC strategy. For example, in most KITTI sub datasets, an important amount of *cars* were labeled as *buildings* by the baseline (No Fusion) whereas they are properly labeled with our JTSC approach. To a lesser extent, the same is also observed for *sidewalks* and *grounds*.

Gradient Reversal (GR) and Dataset Equilibrium (DE). As explained in Section 3, to make the best of datasets with different distributions, taking the correlations between labels into account might not be enough to obtain better results (compared to our baseline). We have thus combined gradient reversal techniques with the joint training approaches presented before to force the network to learn features that are invariant to the dataset. Using gradient reversal for the KITTI dataset does not make sense since the sub-datasets all come from the same distribution. The results of Table 1 shows that adding this gradient reversal does not always improve the performance (e.g., when learning with SIFTFlow and Stanford, the performance measures are worse for the *JT+GR* row than for the *JT* row). The issue tends to happen when one dataset is significantly bigger than the other (7 KITTI > SIFTFlow > Stanford). We thus weighted the contribution of each patch on the gradient computation depending on the size of the dataset this patch come from. The results show the importance of this *Dataset Equilibrium* step even for the KITTI dataset alone. The best results are obtained when the joint training approach is combined both with Dataset Equilibrium and with Gradient Reversal.

4.4. Application to Hand Pose Estimation

To assess the generality of our approach, we have also applied it to a regression problem, namely hand pose estimation from depth images. In this configuration, the coordinates of a set of hand joints must be estimated for each input image, where the number of joints and their geometric position can differ. We used the NYU Hand Pose Dataset [44] and the ICVL Hand Posture

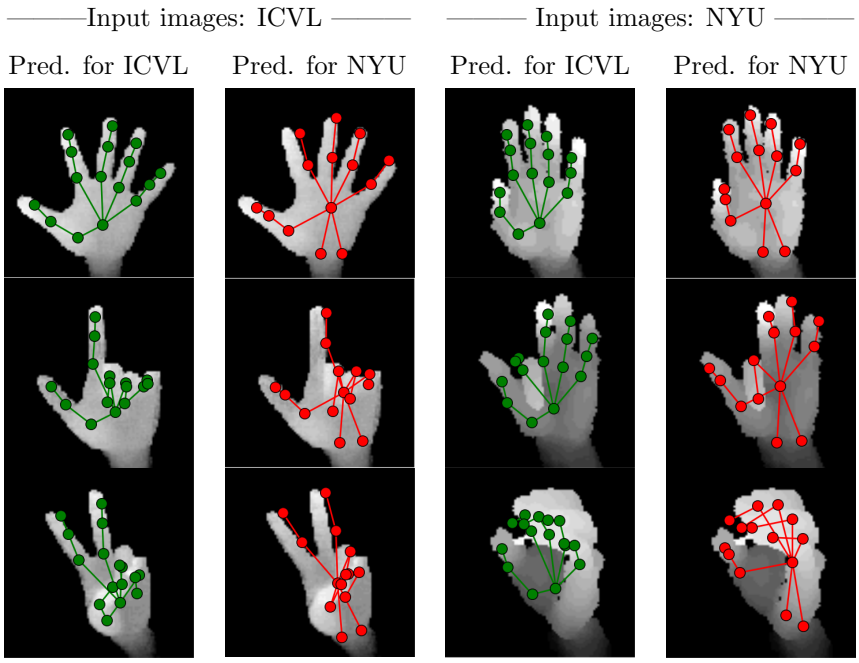


Figure 5: Results of our JTSC strategy applied to the Hand Pose estimation task. The left two columns are taken from ICVL dataset [6], the right two columns are taken from the NYU dataset [44].

Dataset [6]. The first one contains 180k images from 10 subjects, the second one 70k images from a single subject (captured from 3 different view points). The ICVL dataset is annotated with 16 key points in half-VGA images corresponding to *centers* of hand segments (so, strictly speaking not joints), while the NYU dataset is typically used for estimating positions of 14 *joint* positions in VGA images. The semantic meaning of the labels (and so the task) is therefore different, as the geometric positions of the keypoints on the hand surface are different (16 finger centers vs. 14 finger joints).

Inspired from [45], we used three convolutional layers with $32 \times 3 \times 3$ filters (where the first and second layers are followed by a 2×2 max pooling) and three fully connected layers with 1200 hidden units each. The network is trained to minimize the L2 error using a Stochastic Gradient Descent and batches of 100 samples composed of 50 NYU and 50 ICVL samples. We added dropout with a

	No Fusion		Joint Training		JTSC	
	2D, px	3D, mm	2D, px	3D, mm	2D, px	3D, mm
NYU	8.49	17.75	8.16(-3.81%)	17.11(-3.63%)	8.02(-5.47%)	16.80(-5.37%)
ICVL	4.76	9.53	4.62(-2.82%)	9.38(-2.57%)	4.56(-4.13%)	9.16(-3.92%)
Both	N/A	16.42	N/A	15.85(-3.43%)	N/A	**15.56(-5.23%)**

Table 6: Results on regression for a hand pose estimation task on 2 datasets: NYU Dataset [44] and ICVL Dataset [6]. We report the 2D pixel error and the 3D error in mm. Smaller values are better, negative percentage means improvement. Combining pixel error over datasets is not meaningful, as resolutions are different.

drop factor of 10% for all fully connected layers. The JTSC model features add an additional layer with 300 hidden units.

As we can see from Table 6, the selective loss is also helpful in the regression case. Without context, the more complex dataset (NYU) benefits from the additional data, the overall error being smaller. Adding shared context makes training beneficial for both tasks. The gradient reversal was not helpful on this application. We believe that this is due to the similarity of depth images between tasks compared to RGB images.

Table 7 gives the performance of this task compared to the state-of-the-art. We can see that the 3D error of our method is very competitive, giving the best performance apart from the method in [45], which uses additional synthetic data. We explain the good performance of our method with additional time we spent on careful optimization of the baselines by tuning the architecture and the training regime. This appeared to be crucial, in particular the choice of batch normalization [47] and Adam optimization [48]. Before that, we were unable to reproduce the results in [46].

Figure 5 visually illustrates the performances of the methods on images from the two datasets and using predictions targeting the two different datasets. We can see that the label definition (number and geometric positions of the hand keypoints) differ significantly between the two sets.

Method	NYU (3D, mm)	ICVL (3D, mm)
Ours, no fusion	17.76	9.54
Ours, joint training	17.11	9.39
Ours, JTSC	16.80	9.16
Neverova et al. [45]	14.94	-
DeepPrior, Oberweger et al. [46]	19.8†	9.6†
Tompson et al. [44]	21.0†⋆	-
Multi-Scale, Oberweger et al. [46]	27.5†	11.7†
Deep, Oberweger et al. [46]	30.5†	11.8†
Shallow, Oberweger et al. [46]	34.5†	11.7†
Tang et al. [6]	-	12.6†⋆

Table 7: Comparison with state-of-the-art methods of joint position estimation on NYU and ICVL datasets. † values were estimated from plots if authors do not provide numerical values; ⋆ performance was reported in [46].

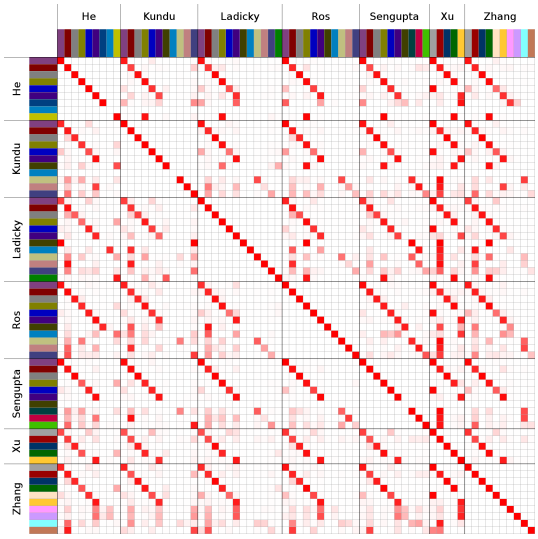


Figure 6: Empirical Label correlation matrix. Each line corresponds to the average of the predictions of the network for a given target class (among the 68 labels given in Fig. 3). Darker cells indicate higher probabilities. Non-diagonal red cells correspond to labels highly correlated with the target (main diagonal) label.

4.5. Detailed analysis on correlations across tasks

Directly leveraging the correlations of the different labelsets over tasks is beneficial, which we showed on both applications, semantic segmentation and pose regression. For the classification problem, we show that these correlations

are indeed meaningful. We computed a label correlation matrix for all sub-datasets of the KITTI dataset, shown in Figure 6, by averaging the predictions made by the network for each target class label (from one of the 7 possible labelings). The (full) diagonal of the matrix gives the correlation rates between the expected target labels. In each line, the other non-zero values correspond to the labels correlated with the target label y_i^k. We can see that a diagonal is visible *inside* each block, including the *off-diagonal blocks*, in particular for the first 5 labels of each block. This means that, as expected, these first 5 labels are all correlated across datasets. For instance, the label *Road* from He et al. is correlated with the label *Road* from Kundu et al. with the *Road* from Ladicky et al. etc. A second observation is that the correlation matrix is not symmetric. For example, the classes *Building*, *Poles*, *Signage* and *Fence* from Sengupta et al. have (as already discussed in Section 4.2) a high correlation with the class *Infrastructure* from Xu et al., meaning that these classes overlap. On the contrary, the class *Infrastructure* from Xu et al. has a very high correlation with the class *Building* from Sengupta et al. and a limited one with the classes *Poles*, *Signage* and *Fence*. This is due to the label distributions: the *Building* class from Sengupta et al. is more represented than the three other classes, so *Infrastructure* from Xu et al. is more correlated to *Building*. These observations mostly confirm the expectations we discussed in Section 4.2, they show that our method can also be used to automatically discover correlations between labels.

5. Conclusion

In this paper, we considered the problem of multi-task multi-domain learning: we want to exploit multiple datasets that have related inputs (e.g., images) but that have been annotated for different tasks (here labels). This problem is important in two major situations: to fuse existing (small) datasets and to reuse existing dataset(s) for a related custom (new) task. We introduced a new *selective loss* function for deep networks that makes it possible to learn jointly across different tasks. For more correlated tasks, we propose to use an auto-context

approach that further exploits the task correlations. We provided experimental results on two computer vision tasks (semantic segmentation and hand pose estimation) with a total of 11 datasets (7 subsets of the KITTI Vision benchmark suite and 4 other datasets). The results show that our approach allows to jointly learn from multiple datasets and to outperform per-task learning and classical fine-tuning based approaches. We also show that the domain adaptation methods (gradient reversal) can be applied for multi-task multi-domain learning but needs to be used with care and requires to balance the different datasets.

Acknowledgment

Authors acknowledge the support from the ANR project SoLStiCe (ANR-13-BS02-0002-01) and the ANR project LIVES (ANR-15-CE23-0026-03). They also want to thank Nvidia for providing two Titan X GPU.

References

[1] M. Cordts, M. Omran, S. Ramos, T. Rehfeld, M. Enzweiler, R. Benenson, U. Franke, S. Roth, B. Schiele, The cityscapes dataset for semantic urban scene understanding, in: Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2016.

[2] G. Ros, L. Sellart, J. Materzynska, D. Vazquez, A. Lopez, SYNTHIA: A large collection of synthetic images for semantic segmentation of urban scenes, 2016.

[3] A. Gaidon, Q. Wang, Y. Cabon, E. Vig, Virtual worlds as proxy for multi-object tracking analysis, in: CVPR, 2016.

[4] Y. Guo, Y. Liu, A. Oerlemans, S. Lao, S. Wu, M. S. Lew, Deep learning for visual understanding: A review, Neurocomputing 187 (2016) 27–48.

[5] J. Shotton, A. Fitzgibbon, M. Cook, T. Sharp, M. Finocchio, R. Moore, A. Kipman, A. Blake, Real-time human pose recognition in parts from single depth images, in: CVPR, 2011.

[6] D. Tang, T. Yu, T.-K. Kim, Real-time articulated hand pose estimation using semi-supervised transductive regression forests, in: ICCV, 2013.

[7] N. Neverova, C. Wolf, G. Taylor, F. Nebout, Hand segmentation with structured convolutional learning, in: ACCV, 2014.

[8] S. Gould, R. Fulton, D. Koller, Decomposing a scene into geometric and semantically consistent regions, in: ICCV, 2009.

[9] C. Liu, J. Yuen, A. Torralba, Nonparametric scene parsing via label transfer, IEEE TPAMI (12).

[10] H. He, B. Upcroft, Nonparametric semantic segmentation for 3d street scenes, in: Intelligent Robots and Systems (IROS), 2013.

[11] A. Kundu, Y. Li, F. Dellaert, F. Li, J. M. Rehg, Joint semantic segmentation and 3d reconstruction from monocular video, in: ECCV, 2014.

[12] L. Ladicky, J. Shi, M. Pollefeys, Pulling things out of perspective, in: CVPR, 2014.

[13] G. Ros, S. Ramos, M. Granados, A. Bakhtiary, D. Vazquez, A. M. Lopez, Vision-based offline-online perception paradigm for autonomous driving, in: Winter Conference on Applications of Computer Vision (WACV), 2015.

[14] S. Sengupta, E. Greveson, A. Shahrokni, P. H. Torr, Urban 3d semantic modelling using stereo vision, in: IEEE ICRA, 2013.

[15] P. Xu, F. Davoine, J.-B. Bordes, H. Zhao, T. Denoeux, Information fusion on oversegmented images: An application for urban scene understanding, in: IAPR MVA, 2013.

[16] R. Zhang, S. A. Candra, K. Vetter, A. Zakhor, Sensor fusion for semantic segmentation of urban scenes, in: IEEE ICRA, 2015.

[17] J. Fritsch, J. T. Kuhnl, A. Geiger, A new performance measure and evaluation benchmark for road detection algorithms, in: Intelligent Transportation Systems-(ITSC), 2013.

[18] A. Geiger, P. Lenz, C. Stiller, R. Urtasun, Vision meets robotics: The kitti dataset, The International Journal of Robotics Research.

[19] Y. Ganin, V. Lempitsky, Unsupervised domain adaptation by backpropagation, in: ICML, 2015.

[20] E. Tzeng, J. Hoffman, T. Darrell, K. Saenko, Simultaneous deep transfer across domains and tasks, in: IEEE International Conference on Computer Vision, ICCV 2015, Santiago, Chile, 2015, pp. 4068–4076.

[21] G. Hinton, O. Vinyals, J. Dean, Distilling the knowledge in a neural network, in: NIPS Deep Learning Workshop, 2014.

[22] J. Yosinski, J. Clune, Y. Bengio, H. Lipson, How transferable are features in deep neural networks?, in: NIPS, 2014.

[23] X. Zhang, F. X. Yu, S. Chang, S. Wang, Deep transfer network: Unsupervised domain adaptation, arXiv.
URL http://arxiv.org/abs/1503.00591

[24] H. Nam, B. Han, Learning multi-domain convolutional neural networks for visual tracking, in: The IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2016.

[25] Y. Yang, T. M. Hospedales, A unified perspective on multi-domain and multi-task learning, in: ICLR, 2015.

[26] P. Kontschieder, S. Bulo, M. Pelillo, H. Bischof, Structured labels in random forests for semantic labelling and object detection, ICCV.

[27] C. Farabet, C. Couprie, L. Najman, Y. LeCun, Learning hierarchical features for scene labeling, IEEE TPAMI (8).

[28] P. Pinheiro, R. Collobert, Recurrent convolutional neural networks for scene labeling, in: ICML, 2014.

[29] T. Kekec, R. Emonet, E. Fromont, A. Trémeau, C. Wolf, Contextually constrained deep networks for scene labeling, in: Proceedings of the British Machine Vision Conference, BMVA Press, 2014. doi:http://dx.doi.org/10.5244/C.28.59.

[30] A. Sharma, O. Tuzel, M. Liu, Recursive context propagation network for semantic scene labeling, in: Advances in Neural Information Processing Systems 27: Annual Conference on Neural Information Processing Systems, Montreal, Canada, 2014, pp. 2447–2455.

[31] A. Sharma, O. Tuzel, D. W. Jacobs, Deep hierarchical parsing for semantic segmentation, in: IEEE Conference on Computer Vision and Pattern Recognition, CVPR Boston, USA, 2015, pp. 530–538.

[32] J. Tighe, S. Lazebnik, Superparsing: Scalable nonparametric image parsing with superpixels, in: ECCV, 2010.

[33] Z. Tu, X. Bai, Auto-context and its application to high-level vision tasks and 3d brain image segmentation, IEEE TPAMI (10).

[34] R. Shapovalov, D. Vetrov, P. Kohli, Spatial inference machines, in: CVPR, 2013.

[35] A. Torralba, K. P. Murphy, W. T. Freeman, Sharing features: Efficient boosting procedures for multiclass object detection, in: CVPR (2), 2004. doi:10.1109/CVPR.2004.232.
URL http://doi.ieeecomputersociety.org/10.1109/CVPR.2004.232

[36] A. Bearman, O. Russakovsky, V. Ferrari, L. Fei-Fei, What's the point: Semantic segmentation with point supervision, arXiv:1506.02106.

[37] P. Pinheiro, R. Collobert, From image-level to pixel-level labeling with convolutional networks, in: CVPR, 2015.

[38] D. Fourure, R. Emonet, É. Fromont, D. Muselet, A. Trémeau, C. Wolf, Semantic segmentation via multi-task, multi-domain learning, in: Proceed-

ings of Structural, Syntactic, and Statistical Pattern Recognition (SSPR), Mérida, Mexico, 2016, pp. 333–343.

[39] M. Zeiler, R. Fergus, Visualizing and understanding convolutional networks, in: ECCV, 2014.

[40] R. Collobert, K. Kavukcuoglu, C. Farabet, Torch7: A matlab-like environment for machine learning, in: BigLearn, NIPS Workshop, 2011.

[41] A. Krizhevsky, I. Sutskever, G. Hinton, Imagenet classification with deep convolutional neural networks, in: NIPS, 2012.

[42] S. Ioffe, C. Szegedy, Batch normalization: Accelerating deep network training by reducing internal covariate shift, in: ICML, 2015.

[43] G. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever, R. Salakhutdinov, Improving neural networks by preventing co-adaptation of feature detectors, arxiv:1207.0580.

[44] J. Tompson, M. Stein, Y. Lecun, K. Perlin, Real-time continuous pose recovery of human hands using convolutional networks, ACM Trans. Graph. (5) (2014) 169:1–169:10.

[45] N. Neverova, C. Wolf, G. Taylor, F. Nebout, Hand pose estimation through semi-supervised and weakly-supervised learning, in: arXiv:1511.06728, 2015.

[46] M. Oberweger, P. Wohlhart, V. Lepetit, Hands deep in deep learning for hand pose estimation, in: CVWW, 2015.

[47] S. Ioffe, C. Szegedy, Batch normalization: Accelerating deep network training by reducing internal covariate shift, 2015.

[48] D. Kingma, J. Ba, Adam: A method for stochastic optimization, 2015.