
Electronic Theses and Dissertations, 2020-

2020

Learning Transferable Representations for Visual Recognition

Yang Zhang
University of Central Florida



Part of the [Computer Sciences Commons](#)

Find similar works at: <https://stars.library.ucf.edu/etd2020>

University of Central Florida Libraries <http://library.ucf.edu>

This Doctoral Dissertation (Open Access) is brought to you for free and open access by STARS. It has been accepted for inclusion in Electronic Theses and Dissertations, 2020- by an authorized administrator of STARS. For more information, please contact STARS@ucf.edu.

STARS Citation

Zhang, Yang, "Learning Transferable Representations for Visual Recognition" (2020). *Electronic Theses and Dissertations, 2020-*. 162.

<https://stars.library.ucf.edu/etd2020/162>

LEARNING TRANSFERABLE REPRESENTATIONS FOR VISUAL RECOGNITION

by

YANG ZHANG
M.S. University of Central Florida, 2016

A dissertation submitted in partial fulfilment of the requirements
for the degree of Doctor of Philosophy
in the Department of Computer Science
in the College of Engineering and Computer Science
at the University of Central Florida
Orlando, Florida

Spring Term
2020

Major Professor: Hassan Foroosh

© 2020 Yang Zhang

ABSTRACT

In the last half-decade, a new renaissance of machine learning originates from the applications of convolutional neural networks to visual recognition tasks. It is believed that a combination of big curated data and novel deep learning techniques can lead to unprecedented results. However, the increasingly large training data is still a drop in the ocean compared with scenarios in the wild. In this literature, we focus on learning transferable representation in the neural networks to ensure the models stay robust, even given different data distributions. We present three exemplar topics in three chapters, respectively: zero-shot learning, domain adaptation, and generalizable adversarial attack. By zero-shot learning, we enable models to predict labels not seen in the training phase. By domain adaptation, we improve a model's performance on the target domain by mitigating its discrepancy from a labeled source model, without any target annotation. Finally, the generalization adversarial attack focuses on learning an adversarial camouflage that ideally would work in every possible scenario. Despite sharing the same transfer learning philosophy, each of the proposed topics poses a unique challenge requiring a unique solution. In each chapter, we introduce the problem as well as present our solution to the problem. We also discuss some other researchers' approaches and compare our solution to theirs in the experiments.

TABLE OF CONTENTS

LIST OF FIGURES	ix
LIST OF TABLES	xiv
CHAPTER 1: INTRODUCTION	1
CHAPTER 2: LITERATURE REVIEW	4
2.1 Image Tagging	4
2.2 Word Embedding	4
2.3 Zero-shot Learning	5
2.4 Domain adaptation	5
2.5 Semantic Segmentation	6
2.6 Domain Adaptation for Semantic Segmentation	7
2.7 Adversarial Attack and its Generalization	8
2.8 Simulation aided Machine Learning	9
CHAPTER 3: ZERO SHOT IMAGE TAGGING	10
3.1 Problem Introduction	10

3.2	The linear rank-ability of word vectors	14
3.2.1	The regulation over words due to image tagging	14
3.2.2	Principal direction and cluster structure	15
3.2.3	Testing the linear rank-ability hypothesis	16
3.3	Approximating the linear ranking functions	19
3.3.1	Image tagging by ranking	19
3.3.2	Approximation by linear regression	20
3.3.3	Approximation by neural networks	21
3.4	Experiments on NUS-WIDE	23
3.4.1	Dataset and evaluation	23
3.4.2	Evaluation	24
3.4.3	Conventional image tagging	25
3.4.4	Zero-shot and Seen/Unseen image tagging	27
3.4.5	Predicting images with 4,093 unseen tags	30
3.4.6	Qualitative results	31
3.5	Experiments on IAPRTC-12	31
3.5.1	Configuration	31

3.5.2	Results	33
3.5.3	Qualitative results	34
3.6	Summary	35
CHAPTER 4: DOMAIN ADAPTATION FOR SEMANTIC SEGMENTATION		36
4.1	Problem Introduction	36
4.2	Approach	41
4.2.1	Preliminaries	41
4.2.2	Domain adaptation observing the target properties	42
4.2.3	Inferring the target properties	44
4.2.4	Curriculum domain adaptation: recapitulation	47
4.2.5	Color Constancy	48
4.3	Experiments	51
4.3.1	Segmentation network and optimization	52
4.3.2	Datasets and evaluation	53
4.3.3	Results of inferring global label distribution	55
4.3.4	Domain adaptation experiments	56
4.3.5	Representations of the superpixels	63

4.3.6	Granularity of the superpixels	65
4.3.7	Domain adaptation experiments using ADEMAPP	66
4.3.8	What is the “market value” of the synthetic data?	67
4.4	Review of the recent works on domain adaptation for semantic segmentation	69
4.4.1	Adversarial training based methods	69
4.4.2	Other methods	72
4.4.3	Results reported in other papers	73
4.4.4	The existing methods and ours are complementary	73
4.5	Summary	76
CHAPTER 5: LEARNING TRANSFERABLE ADVERSARIAL CAMOUFLAGES		78
5.1	Problem Introduction	78
5.2	Learning Objective	81
5.3	Approach	83
5.3.1	Sampling transformations to estimate \mathbb{E}_t	83
5.3.2	Learning a clone network $V_\theta(c, t)$ to approximate $V_t(c)$	84
5.3.3	Jointly learning the clone network and the optimal camouflage	86
5.4	Experiments	87

5.4.1	Experiment setup	88
5.4.2	Resolution of the camouflages	92
5.4.3	Camouflaging Toyota Camry in the urban environment	93
5.4.4	Virtual SUV in urban area	94
5.4.5	Transferability across detectors	95
5.4.6	Transferability across environments	96
5.4.7	Transferability across vehicles	97
5.4.8	Transferability across viewing positions	98
5.4.9	Impact of clone network quality	99
5.4.10	Detection attention	100
5.4.11	Simulation implementation	102
5.4.12	Simulation error	103
5.4.13	Non-linearity and non-convexity	103
5.4.14	Qualitative results	104
5.5	Summary	105
CHAPTER 6: CONCLUSION		107
LIST OF REFERENCES		108

LIST OF FIGURES

3.1	Given an image, its relevant tags' word vectors rank ahead of the irrelevant tags' along some direction in the word vector space. We call that direction the principal direction for the image. To solve the problem of image tagging, we thus learn a function $f(\cdot)$ to approximate the principal direction from an image. This function takes as the input an image x_m and outputs a vector $f(x_m)$ for defining the principal direction in the word vector space.	10
3.2	Visualization of the offsets between relevant tags' word vectors and irrelevant ones'. <i>Note that each vector from the origin to a point is an offset between two word vectors.</i> The relevant tags are shown beside the images [39].	14
3.3	The existence (left) and generalization (right) of the principal direction for each visual association rule in words induced by an image.	17
3.4	The neural network used in our approach for implementing the mapping function $f(x; \theta)$ from the input image, which is represented by the CNN features x , to its corresponding principal direction in the word vector space.	21
3.5	The top five tags for exemplar images [39] returned by Fast0Tag on the conventional, zero-shot, and seen/unseen image tagging tasks, and by TagProp for conventional tagging. (Correct tags: green ; mistaken tags: red and <i>italic</i> . Best viewed in color.)	30

3.6	The top five tags for exemplar images in [39](a) and [79](b) returned by Fast0Tag on the conventional, zero-shot, seen/unseen and 4,093 zero-shot image tagging tasks, and by TagProp for conventional tagging. (Correct tags: green ; mistaken tags: red and <i>italic</i>)	34
4.1	The overall framework of our curriculum domain adaptation approach to the semantic segmentation of urban scenes.	47
4.2	Predictions by the same FCN-8s model, without domain adaptation, before and after we calibrate the image’s colors.	48
4.3	Qualitative semantic segmentation results on the Cityscapes dataset [158] (target domain). For each target image in the first column, we retrieve its nearest neighbor from the SYNTHIA [40] dataset (source domain). The third column plots the label distributions due to the groundtruth pixel-wise semantic annotation, the predictions by the baseline network with no adaptation, and the inferred distribution by logistic regression. The last three columns are the segmentation results by the baseline network, our domain adaptation approach, and human annotators, respectively.	50

4.4	Qualitative semantic segmentation results on the Cityscapes dataset [158] (target domain). For each target image in the first column, we retrieve its nearest neighbor from the GTA [156] dataset (source domain). The third column plots the label distributions due to the groundtruth pixel-wise semantic annotation, the predictions by the baseline network with no adaptation, and the inferred distribution by logistic regression. The last three columns are the segmentation results by the baseline network, our domain adaptation approach, and human annotators, respectively.	51
4.5	Confusion matrices for the baseline of no adaptation (left) and Ours (CC+I+SP) (right) for the experiments of SYNTHIA-to-Cityscapes (SYNTHIA2Cityscapes, top) and GTA-to-Cityscapes (GTA2Cityscapes, bottom).	62
4.6	Some “train” and “bus” images from the Cityscapes and GTA datasets. We can see that the Cityscapes “trains” are visually more similar to the GTA “buses” than to the GTA “trains”.	63
4.7	We evaluate how many superpixels are accurate in the top $x\%$ confidently predicted superpixels. The experiments are conducted on the validation set of Cityscapes with color constancy.	66
4.8	Pairwise comparison between different domain adaptation methods for the semantic segmentation task. The entry (i, j) of this table is the number of classes by which the i -th method outperforms the j -th. The results are obtained on GTA2Cityscapes. Our method is labeled <i>Curriculum</i>	76

5.1	A Toyota Camry XLE in the center of the image fools the Mask R-CNN object detector after we apply the learned camouflage to it (on the right), whereas neither plain colors (on the left) nor a random camouflage (in the middle) is able to escape the Camry from being detected.	79
5.2	Example procedure of a transformation t . From left to right: a 16×16 camouflage, a high-fidelity vehicle, a location in our simulated environment, and an image due to this transformation over the camouflage.	82
5.3	The camera setup as well as some exemplar locations. The cameras depicted in the green color are used to learn the camouflage while those in red are unseen cameras used for testing the camouflage’s generalization. (H: camera height, L: vehicle-to-camera distance.)	83
5.4	Overview of our optimization pipeline and the clone network $V_{\theta}(c, t)$	86
5.5	The two environments we built to learn and test the vehicle camouflage. Zoom in for more details. These images are of the higher resolution but the same rendering quality (anti-aliasing level, shadow quality, rendering distance, texture resolution etc.) the detector perceived.	88
5.6	A fraction of different Toyota sedan appearances in MS COCO dataset. Their appearances are diverse.	90
5.7	Orthogonal views of the Toyota Camry XLE 2015 (second row) and the virtual SUV (first row) used in our simulation.	90

5.8	The mIoU and P@0.5 of the 800 random camouflages in different resolutions on Camry in the DownTown environment. There are 100 camouflages per resolution.	93
5.9	Visualization of three random camouflages of resolutions 4×4 , 16×16 , and 256×256 , respectively, as well as the resulting images by the same camera.	93
5.10	Clone network’s learned camouflage’s classification score and mIoU vs. Simulation called. We can see how the new samples helped the clone network to find the minimal.	100
5.11	The best detections in each image and the gradient heatmap of their classification scores w.r.t. the input images. The detector places its attention predominantly on the upper car body, i.e., roof, hood, trunk, and windows.	102
5.12	Qualitative comparison of the Mask R-CNN detections results of the grey baseline color, random camouflages and our learned camouflages in different transformations. Zoom in for more details.	104

LIST OF TABLES

3.1	Comparison results of the conventional image tagging with 81 tags on NUS-WIDE.	26
3.2	Comparison results of the zero-shot and seen/unseen image tagging tasks with 81 unseen tags and 925 seen tags.	26
3.3	Annotating images with up to 4,093 unseen tags.	29
3.4	Comparison results of the conventional image tagging with 291 tags on IAPRTC-12.	32
3.5	Comparison results of the zero-shot and seen/unseen image tagging tasks with 58 unseen tags and 233 seen tags on IAPRTC-12.	32
4.1	χ^2 distances between the groundtruth label distributions and those predicted by different methods for the adaptation from SYNTHIA to Cityscapes.	55
4.2	Comparison results for adapting the FCN-8s model from SYNTHIA to Cityscapes.	58
4.3	Comparison results for adapting the FCN-8s model from GTA to Cityscapes.	58
4.4	Results for the adaptation of FCN-8s from GTA to Cityscapes when we use handcrafted features instead of the CNN features.	63
4.5	Results for the adaptation of FCN-8s from GTA to Cityscapes when we use different numbers of superpixels per image. Here the images are pre-processed with color constancy.	66

4.6	Results for the adaptation of ADEMXPAPP [199] from GTA to Cityscapes. The ADEMXPAPP net is a more powerful semantic segmentation network than FCN-8s.	67
4.7	IoUs after mixing different percentages of Cityscapes images into the SYNTHIA training dataset (SYN+CS), and of models trained with different percentages of Cityscapes images without any SYNTHIA images (CS only). . .	68
4.8	Comparison with the recent works published after the conference version of our approach [210].	75
5.1	Mask R-CNN detection performance on the Camry in the urban environment.	94
5.2	Detection performance of camouflages on SUV in urban environment.	95
5.3	YOLOv3-SPP detection performance on the Camry in the urban environment. In addition to the camouflage inferred for YOLO, we also include the YOLO detection results on the camouflage learned for Mask R-CNN.	96
5.4	Detection performance of camouflages on Camry in Landscape environment. Note that this camouflage is pretrained in urban environment and then transferred without any finetuning.	97
5.5	Camouflage transferability across vehicle reported in testing P@0.5 in urban environment.	98
5.6	Detection performance of pretrained camouflages on Camry with urban environment in 16 unseen cameras.	99

CHAPTER 1: INTRODUCTION

Transfer learning focuses on applying learned knowledge on a different and yet related problem. It has been accompanying machine learning for a long time. Some topics of transfer learning include domain adaptation, transductive learning, zero-shot learning, and semi-supervised learning, etc. Transfer learning has been an active research field before the rise of deep learning.

However, it was not until neural network regained attention [99] did researchers realize there are ample new opportunities. The neural networks bring two new challenges to the table for transfer learning: feature learning and complex tasks. One of the main reasons that neural networks being popular is their learning powerful and transferable intermediate features [205] directly from high dimension raw input. The capability of learning representations from raw data encourages researchers to take advantage of the feature learning of the neural networks in transfer learning. Take domain adaptation as an example. Previous shallow domain adaptation [143] aims to adapt machine learning models with given features. In contrast, recent deep domain adaptation [65] concentrates on learning CNN as an invariant feature extractor with raw input. How will we train a neural network as a high-performing and robust feature extractor?

Secondly, increasing neural network capability benefits many structured prediction tasks in an end-to-end style. For example, object detection used to need a dedicated complex model such as DPM [53] and a considerable amount of tricks to train the model. Nowadays, one Mask-RCNN [81] is sufficient. Such advancement paves the way for studying previously difficult tasks in the scope of transfer learning. For instance, domain adaptation for segmentation [86, 210] followed FCN [114]. However, the more complex the tasks become, the harder it is to disentangle features in transfer learning. How can we disentangle the features in complex tasks and networks?

After the literature review chapter, we present three diverse yet relevant topics to showcase how

deep learning has fundamentally changed different transfer learning problems and how we can use deep learning as a tool to address the previous two questions.

We first describe zero-shot image tagging in chapter 3. Zero-shot learning trains a model to predict labels that are unseen during the training phase. It is common to transfer knowledge of known training labels to unseen labels via their semantically encoded word vectors. However, unlike commonly studied zero-shot image classification, we want to study a more general image tagging scenario, where each image may possess multiple labels. We begin by looking into a particular image-word relation in this chapter. Our results show that the word vectors of relevant tags for a given image rank ahead of the irrelevant tags, along a principal direction in the word vector space. Inspired by this observation, we propose to solve image tagging by estimating the principal direction for an image. Particularly, we exploit linear mappings and nonlinear deep neural networks to approximate the principal direction from an input image. We arrive at a quite versatile tagging model. It runs fast given a test image, in constant time w.r.t. the training set size. It not only yields superior performance for the conventional tagging task on the NUS-WIDE dataset, but also outperforms competitive baselines on annotating images with previously **unseen** tags.

In chapter 4, we describe domain adaptation for semantic segmentation and our solution to it. Training semantic segmentation CNNs requires a considerable amount of data, which is difficult to collect and laborious to annotate. Recent advances in computer graphics make it possible to train CNNs on photo-realistic synthetic imagery with computer-generated annotations. Despite this, the domain mismatch between real images and the synthetic data hinders the models' performance. Hence, we propose a curriculum-style learning approach to minimizing the domain gap in urban scene semantic segmentation. The curriculum domain adaptation solves easy tasks first to infer necessary properties about the target domain; in particular, the first task is to learn global label distributions over images and local distributions over landmark superpixels. These are easy to estimate because images of urban scenes have strong idiosyncrasies (e.g., the size and spatial

relations of buildings, streets, cars, etc.). We then train a segmentation network, while regularizing its predictions in the target domain to follow those inferred properties. In experiments, our method outperforms the baselines on two datasets and three backbone networks. We also report extensive ablation studies about our approach.

Lastly, in chapter 5, we conduct an intriguing experimental study about a generalizable physical adversarial attack on object detectors in the wild. In particular, we learn a camouflage pattern to hide vehicles from being detected by state-of-the-art convolutional neural network based detectors. Our approach alternates between two threads. In the first, we train a neural approximation function to imitate how a simulator applies a camouflage to vehicles and how a vehicle detector performs given images of the camouflaged vehicles. In the second, we minimize the approximated detection score by searching for the optimal camouflage. Experiments show that the learned camouflage can not only hide a vehicle from the image-based detectors under many test cases but also generalizes to different environments, vehicles, and object detectors.

CHAPTER 2: LITERATURE REVIEW

2.1 Image Tagging

Image tagging aims to assign relevant tags to an image or to return a ranking list of tags. In the literature this problem has been mainly approached from the tag ranking perspective. In the generative methods, which involve topic models [17, 125, 204, 133] and mixture models [104, 93, 182, 54, 29, 46], the candidate tags are naturally ranked according to their probabilities conditioned on the test image. For the non-parametric nearest neighbor based methods [120, 121, 108, 95, 80, 107, 214], the tags for the test image are often ranked by the votes from some training images. The nearest neighbor based algorithms, in general, outperform those depending on generative models [95, 109], but suffer from high computation costs in both training and testing. The recent FastTag algorithm [34] is magnitude faster and achieves comparable results with the nearest neighbor based methods. The embedding method [196] assigns ranking scores to the tags by a cross-modality mapping between images and tags. This idea is further exploited using deep neural networks [75]. Interestingly, none of these methods learn their models explicitly for the ranking purpose except [196, 75], although they all rank the candidate tags for the test images. Thus, there exists a mismatch between the models learned and the actual usage of the models, violating the principle of Occam's razor.

2.2 Word Embedding

Instead of representing words using the traditional one-hot vectors, word embedding maps each word to a continuous-valued vector, by learning from primarily the statistics of word co-occurrences. Although there are earlier works on word embedding [162, 45], we point out that our work focuses

on the most recent GloVe [148] and word2vec vectors [124, 123, 122]. As shown in the well-known word analogy experiments [124, 148], both types of word vectors are able to capture fine-grained semantic and syntactic regularities using vector offsets.

2.3 Zero-shot Learning

Zero-shot learning is often used exchange-ably with zero-shot classification, whereas the latter is a special case of the former. Unlike weakly-supervised learning [128, 59] which learn new concepts by mining noisy new samples, zero-shot classification learns classifiers from seen classes and aims to classify the objects of unseen classes [136, 135, 103, 6, 60, 92, 135, 136, 177]. Attributes [102, 52] and word vectors are two of the main semantic sources making zero-shot classification feasible.

Our Fast0Tag along with [61] enriches the family of zero-shot learning by zero-shot multi-label classification [189]. Fu et al. [61] reduce the problem to zero-shot classification by treating every combination of the multiple labels as a class. We instead directly model the labels and are able to assign/rank many unseen tags for an image.

2.4 Domain adaptation

Conventional machine learning algorithms rely on the standard assumption that the training and test data are drawn i.i.d. from the same underlying distribution. However, it is often the case that there exists some discrepancy between the training and test stages. Domain adaptation aims to rectify this mismatch and tune the models toward better generalization at the test stage [186, 185, 73, 97, 78].

The existing work on domain adaptation mostly focuses on classification and regression prob-

lems [143, 138], e.g., learning from online images to classify real world objects [163, 74], and, more recently, aims to improve the adaptability of deep neural networks [115, 66, 65, 190, 22]. Among them, the most relevant works to ours are those exploring simulated data [179, 203, 158, 194, 86, 146, 172]. Sun and Saenko train generic object detectors from synthetic images [179], while Vazquez et al. use virtual images to improve pedestrian detections in real environments [194]. The other way around, i.e., how to improve the quality of the simulated images using the real ones, is studied in [172, 146].

2.5 Semantic Segmentation

Semantic segmentation is the task of assigning an object label to each pixel of an image. Traditional methods [171, 184, 208] rely on local image features manually designed by domain experts. After the pioneering works [32, 114] that introduced the convolutional neural network (CNN) [106] to semantic segmentation, most recent top-performing methods are also built on CNNs [199, 160, 15, 212, 134, 44].

Currently, there are multiple and increasing numbers of semantic segmentation datasets aiming for different computer vision applications. Some general ones include the PASCAL VOC2012 Challenge [48], which contains nearly 10,000 annotated images for the segmentation competition, and the MS COCO Challenge [111], which includes over 200,000 annotated images. In our paper, we focus on urban outdoor scenes. Several urban scene segmentation datasets are publicly available such as Cityscapes [40], a vehicle-centric dataset created primarily in German cities, KITTI [68], another vehicle-centric dataset captured in the German city Karlsruhe, Berkeley DeepDrive Video Dataset [202], a dashcam dataset collected in United States, Mapillary Vistas Dataset [131], so far known as the largest outdoor urban scene segmentation dataset collected from all over the world, WildDash, a much smaller yet diverse dataset for benchmark purpose, and CamVid [24], a small

and low-resolution toy dashcam dataset. An enormous amount of labor-intensive work is required to annotate the images that are needed to obtain accurate segmentation models. According to [156], it took about 60 minutes to manually segment each image in [23] and about 90 minutes for each in [40]. A plausible approach to reducing the human annotation workload is to utilize weakly supervised information such as image labels and bounding boxes [144, 87, 140, 151].

We instead explore the almost labor-free labeled virtual images for training high-quality segmentation networks. In [156], annotating a synthetic image took only 7 seconds on average through a computer game. For the urban scenes, we use the SYNTHIA [158] and GTA [156] datasets which contain images of virtual cities. Although not used in our experiments, another synthetic segmentation dataset worth mentioning is Virtual KITTI [62], a synthetic duplication of the original KITTI [68] dataset.

2.6 Domain Adaptation for Semantic Segmentation

Due to the clear visual mismatch between synthetic and real data [169, 156, 158], we expect to use domain adaptation to enhance the segmentation performance on real images by networks trained on synthetic imagery. To the best of our knowledge, our work [210] and the FCNs in the wild [86] are among the very first attempts to tackle this problem. It subsequently became an independent track in the Visual Domain Adaptation Challenge (VisDA) 2017 [147]. After that, some feature-alignment based methods were developed [167, 85, 165, 37, 198]. We postpone further discussion to Section 4.4, which presents a comprehensive survey of the works published after ours and before December 2018. We group them into two major categories and describe their main methods. We also summarize their results in Table 4.8 and analyze their complementary relationships with ours.

2.7 Adversarial Attack and its Generalization

Currently, the adversarial attack is powerful enough to attack image classification [126], object detection [11], semantic segmentation [201], audio recognition [28] and even bypass most of the defense mechanism [12]. The mainstream adversarial machine learning research focuses on the in silico ones or the generalization within in silico [113]. Such learned perturbations are practically unusable in the real world as shown in the experiments by [118], who found that almost all perturbation methods failed to prevent a detector from detecting real stop signs.

The first physical world adversarial attack by [101] found perturbations remain effective on the printed paper. [51] found a way to train perturbations that remain effective against a classifier on real stop signs for different viewing angles and such. [13] trained another perturbation that successfully attacks an image classifier on 3D-printed objects. However [119] found that [51]'s perturbation does not fool the object detectors YOLO9000 [152] and Faster RCNN [154]. They argue that fooling an image classifier is different and easier than fooling an object detector because the detector is able to propose object bounding boxes on its own. In the meanwhile, [117]'s and [36]'s work could be generalized to attack the stop sign detector in the physical world more effectively. However, all the above methods aim to perturb detecting the stop signs.

Blackbox attack is another relevant topic. Among the current blackbox attack literature, [141] trained a target model substitution based on the assumption that the gradient between the image and the perturbation is available. We do not have the gradient since the simulation is nondifferentiable. [35] proposed a coordinate descent to attack the model. However, we found that our optimization problem is time-consuming, noisy, empirically non-linear and non-convex. Time constraints made this approach unavailable to us since it requires extensive evaluations during coordinate descent. Besides, coordinate descent generally requires precise evaluation at each data-point.

2.8 Simulation aided Machine Learning

Since the dawn of deep learning, data collection has always been of fundamental importance as deep learning model performance is generally correlated with the amount of training data used. Given the sometimes unrealistically expensive annotation costs, some machine learning researchers use synthetic data to train their models. This is especially true in computer vision applications since state-of-the-art computer-generated graphics are truly photo-realistic. [159, 155] proposed synthetic datasets for semantic segmentation. [63] proposed virtual KITTI as a synthetic replica of the famous KITTI dataset [69] for tracking, segmentation, etc. And [193] proposed using synthetic data for human action learning. [209, 21] adapt RL-trained robot grasping models from the synthetic environment to the real environment. [187] trained a detection network using the same Unreal engine that we are using.

CHAPTER 3: ZERO SHOT IMAGE TAGGING

3.1 Problem Introduction

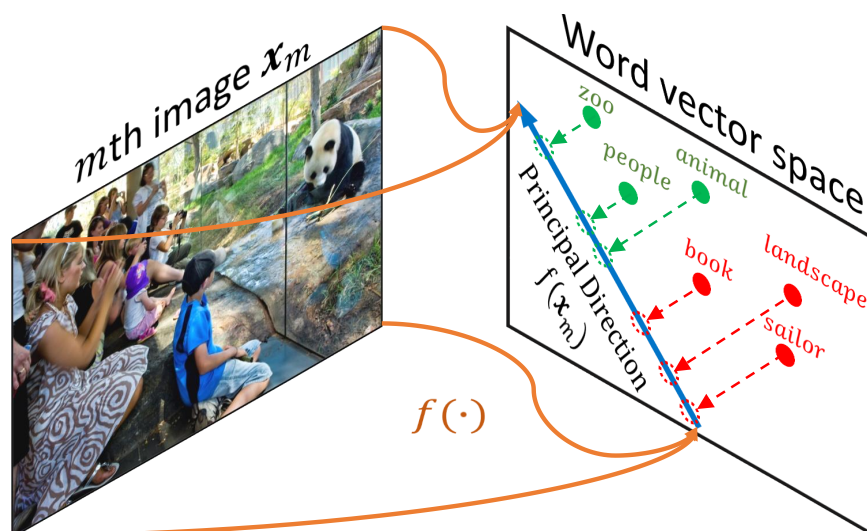


Figure 3.1: Given an image, its relevant tags’ word vectors rank ahead of the irrelevant tags’ along some direction in the word vector space. We call that direction the **principal direction** for the image. To solve the problem of image tagging, we thus learn a function $f(\cdot)$ to approximate the principal direction from an image. This function takes as the input an image x_m and outputs a vector $f(x_m)$ for defining the principal direction in the word vector space.

Recent advances in the vector-space representations of words [122, 123, 148] have benefited both NLP [176, 216, 183] and computer vision tasks such as zeros-shot learning [177, 58, 6] and image captioning [105, 96, 98]. The use of word vectors in NLP is grounded on the fact that the

This chapter contains previously published materials from “Fast zero-shot image tagging” by Yang Zhang, Boqing Gong, and Mubarak Shah, published in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition in 2016.

fine-grained **linguistic** regularities over words are captured by linear word vector offsets—a key observation from the well-known word analogy experiments [124, 148], such as the syntactic relation $dance - dancing \approx fly - flying$ and semantic relation $king - man \approx queen - woman$. However, it is unclear whether the **visual** regularities over words, which are implicitly used in the aforementioned computer vision problems, can still be encoded by the simple vector offsets.

In this chapter, we are interested in the problem of image tagging, where an image (e.g., of a zoo in Figure 3.1) calls for a partition of a vocabulary of words into two disjoint sets according to the image-word relevance (e.g., relevant tags $Y = \{people, animal, zoo\}$ and irrelevant ones $\bar{Y} = \{sailor, book, landscape\}$). This partitioning of words, (Y, \bar{Y}) , is essentially different from the fine-grained syntactic (e.g., dance to dancing) or semantic (e.g., king to man) relation tested in the word analogy experiments. Instead, it is about the relationship between two sets of words due to a visual image. Such a relation in words is semantic and descriptive, and focuses on **visual association**, albeit relatively coarser. In this case, do the word vectors still offer the nice property, that the simple linear vector offsets can depict the visual (image) association relations in words? For the example of the zoo, while humans are capable of easily answering that the words in Y are more related to the zoo than those in \bar{Y} , can such zoo-association relation in words be expressed by the 9 pairwise word vector offsets $\{people - sailor, people - book, \dots, zoo - landscape\}$ between the relevant Y and irrelevant \bar{Y} tags' vectors?

One of the main contributions of this chapter is to empirically examine the above two questions (cf. Section 3.2). Every image introduces a visual association rule (Y, \bar{Y}) over words. Thanks to the large number of images in benchmark datasets for image tagging, we are able to examine many distinct *visual association regulations* in words and the corresponding *vector offsets* in the word vector space. Our results reveal a somehow surprising connection between the two: the offsets between the vectors of the relevant tags Y and those of the irrelevant \bar{Y} are along about the same direction, which we call the **principal direction**. See Figure 3.2 for the visualization of some

vector offsets. In other words, there exists at least one vector (direction) \boldsymbol{w} in the word vector space, such that its inner products with the vector offsets between Y and \bar{Y} are greater than 0, i.e., $\forall \boldsymbol{p} \in Y, \forall \boldsymbol{n} \in \bar{Y}$,

$$\langle \boldsymbol{w}, \boldsymbol{p} - \boldsymbol{n} \rangle > 0 \text{ equivalently, } \langle \boldsymbol{w}, \boldsymbol{p} \rangle > \langle \boldsymbol{w}, \boldsymbol{n} \rangle, \quad (3.1)$$

where the latter reads that the vector \boldsymbol{w} *ranks* all relevant words Y (e.g., for the zoo image) ahead of the irrelevant ones \bar{Y} . For brevity, we overload the notations Y and \bar{Y} to respectively denote the vectors of the words in them.

The visual association relations in words thus represent themselves by the (linear) rank-abilities of the corresponding word vectors. This result reinforces the conclusion from the word analogy experiments that, for a single word multiple relations are embedded in the high dimensional space [124, 148]. Furthermore, those relations can be expressed by simple linear vector arithmetic.

Inspired by the above observation, we propose to solve the image tagging problem by estimating the principal direction, along which the relevant tags rank ahead of the irrelevant ones in the word vector space. Particularly, we exploit linear mappings and deep neural networks to approximate the principal direction from each input image. This is a grand new point of view to image tagging and results in a quite versatile tagging model. It operates fast given a test image, in constant time with respect to the training set size. It not only gives superior performance for the conventional tagging task, but is also capable of assigning novel tags from an open vocabulary, which are unseen at the training stage. We do not assume any *a priori* knowledge about these unseen tags as long as they are in the same vector space as the seen tags for training. To this end, we name our approach fast zero-shot image tagging (Fast0Tag) to recognize that it possesses the advantages of both FastTag [34] and zero-shot learning [103, 60, 61].

In sharp contrast to our approach, previous image tagging methods can only annotate test images with the tags seen at training except [61], to the best of our knowledge. Limited by the static and usually small number of seen tags in the training data, these models are frequently challenged in practice. For instance, there are about 53M tags on Flickr and the number is rapidly growing. The work of [61] is perhaps the first attempt to generalize an image tagging model to unseen tags. Compared to the proposed method, it depends on two extra assumptions. One is that the unseen tags are known *a priori* in order to tune the model towards their combinations. The other is that the test images are known *a priori*, to regularize the model. Furthermore, the generalization of [61] is limited to a very small number, U , of unseen tags, as it has to consider all the 2^U possible combinations.

To summarize, our first main contribution is on the analyses of the visual association relations in words due to images, and how they are captured by word vector offsets. We hypothesize and empirically verify that, for each visual association rule (Y, \bar{Y}) , in the word vector space there exists a principal direction, along which the relevant words' vectors rank ahead of the others'. Built upon this finding, the second contribution is a novel image tagging model, Fast0Tag, which is fast and generalizes to open-vocabulary unseen tags. Last but not least, we explore three different image tagging scenarios: **conventional** tagging which assigns seen tags to images, **zero-shot** tagging which annotates images by (a large number of) unseen tags, and **seen/unseen** tagging which tags images with both seen and unseen tags. In contrast, the existing work tackles either conventional tagging, or zero-shot tagging with very few unseen tags. Our Fast0Tag gives superior results over competitive baselines under all the three testing scenarios.

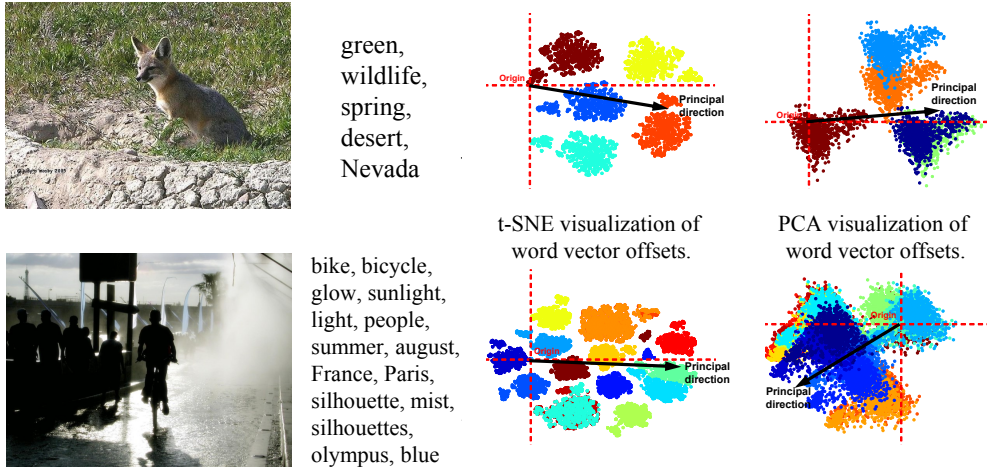


Figure 3.2: Visualization of the offsets between relevant tags’ word vectors and irrelevant ones’. Note that each vector from the origin to a point is an offset between two word vectors. The relevant tags are shown beside the images [39].

3.2 The linear rank-ability of word vectors

Our Fast0Tag approach benefits from the finding that the visual association relation in words, i.e., the partition of a vocabulary of words according to their relevances to an image, expresses itself in the word vector space as the existence of a principal direction, along which the words/tags relevant to the image rank ahead of the irrelevant ones. This section details the finding.

3.2.1 The regulation over words due to image tagging

We use \mathcal{S} to denote the seen tags available for training image tagging models and \mathcal{U} the tags unseen at the training stage. The training data are in the form of $\{(\mathbf{x}_m, Y_m); m = 1, 2, \dots, M\}$, where $\mathbf{x}_m \in \mathbb{R}^D$ is the feature representation of image m and $Y_m \subset \mathcal{S}$ are the seen tags relevant to that image. For brevity, we overload the notation Y_m to also denote the collection of the corresponding

word/tag vectors.

The **conventional** image tagging aims to assign seen tags in \mathcal{S} to the test images. The **zero-shot** tagging, formalized in [61], tries to annotate test images using a pre-fixed set of unseen tags \mathcal{U} . In addition to those two scenarios, this chapter considers **seen/unseen** image tagging, which finds both relevant seen tags from \mathcal{S} and relevant unseen tags from \mathcal{U} for the test images. Furthermore, the set of unseen tags \mathcal{U} could be open and dynamically growing.

Denote by $\overline{Y}_m := \mathcal{L} \setminus Y_m$ the irrelevant seen tags. An image m introduces a visual association regulation to words—the partition (Y_m, \overline{Y}_m) of the seen tags to two disjoint sets. Noting that many fine-grained syntactic and semantic regulations over words can be expressed by linear word vector offsets, we next examine what properties the vector offsets could offer for this new visual association rule.

3.2.2 Principal direction and cluster structure

Figure 3.2 visualizes the vector offsets $(\mathbf{p} - \mathbf{n})$, $\forall \mathbf{p} \in Y_m, \forall \mathbf{n} \in \overline{Y}_m$ using t-SNE [192] and PCA for two visual association rules over words. One is imposed by an image with 5 relevant tags and the other is with 15 relevant tags. We observe two main structures from the vector offsets:

Principal direction. Mostly, the vector offsets point to about the same direction (relative to the origin), which we call the principal direction, for a given visual association rule (Y_m, \overline{Y}_m) in words for image m . This implies that the relevant tags Y_m rank ahead of the irrelevant ones \overline{Y}_m along the principal direction (cf. eq. (3.1)).

Cluster structure. There exist cluster structures in the vector offsets for each visual association regulation over the words. Moreover, all the offsets pointing to the same relevant tag in Y_m

fall into the same cluster. We differentiate the offsets pointing to different relevant tags by colors in Figure 3.2.

Can the above two observations generalize? Namely, do they still hold in the high-dimensional word vector space for more visual association rules imposed by other images? To answer the questions, we next design an experiment to verify the existence of the principal directions in word vector spaces, or equivalently the linear rank-ability of word vectors. We leave the cluster structure for future research.

3.2.3 Testing the linear rank-ability hypothesis

Our experiments in this section are conducted on the validation set (26,844 images, 925 seen tags \mathcal{S} , and 81 unseen tags \mathcal{U}) of NUS-WIDE [39]. The number of relevant seen/unseen tags associated with an image ranges from 1 to 20/117 and on average is 1.7/4.9. See Section 3.4 for details.

Our objective is to investigate, for any visual association rule $(Y_m, \overline{Y_m})$ in words by image m , the existence of the principal direction along which the relevant tags Y_m rank ahead of the irrelevant tags $\overline{Y_m}$. The proof completes once we find a vector \mathbf{w} in the word vector space that satisfies the ranking constraints $\langle \mathbf{w}, \mathbf{p} \rangle > \langle \mathbf{w}, \mathbf{n} \rangle, \forall \mathbf{p} \in Y_m, \forall \mathbf{n} \in \overline{Y_m}$. To this end, we train a linear ranking SVM [94] for each visual association rule using all the corresponding pairs (\mathbf{p}, \mathbf{n}) , then rank the word vectors by the SVM, and finally examine how many constraints are violated. In particular, we employ MiAP, the larger the better (cf. Section 3.4), to compare the SVM’s ranking list with those ranking constraints. We repeat the above process for all the validation images, resulting in 21,863 unique visual association rules.

Implementation of ranking SVM. In this chapter, we use the implementation of solving ranking

SVM in the primal [30] with the following formulation:

$$\min_{\mathbf{w}} \frac{\lambda}{2} \|\mathbf{w}\|^2 + \sum_{\mathbf{y}_i \in Y_m} \sum_{\mathbf{y}_j \in \overline{Y_m}} \max(0, 1 - \mathbf{w}\mathbf{y}_i + \mathbf{w}\mathbf{y}_j)$$

where λ is the hyper-parameter controlling the trade-off between the objective and the regularization.

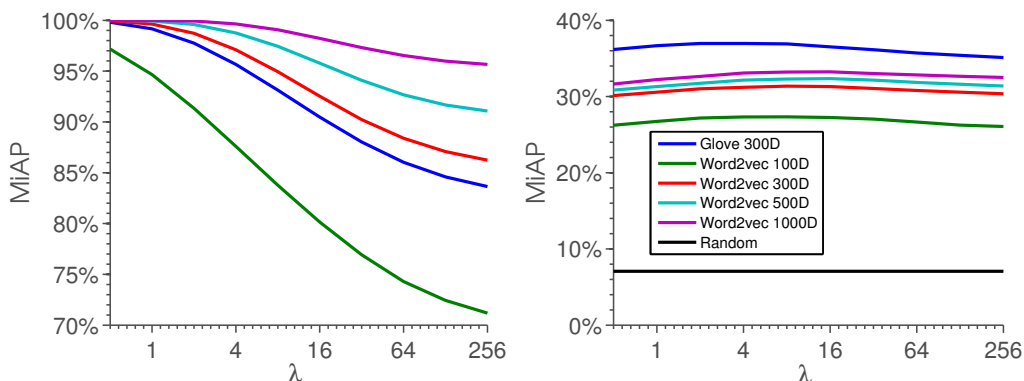


Figure 3.3: The existence (left) and generalization (right) of the principal direction for each visual association rule in words induced by an image.

Results. The MiAP results averaged over all the distinct regularizations are reported in Figure 3.3(left), in which we test the 300D GloVe vectors [148] and word2vec [124] of dimensions 100, 300, 500, and 1000. The horizontal axis shows different regularizations we use for training the ranking SVMs. Larger λ regularizes the models more. In the 300D GloVe space and the word2vec spaces of 300, 500, and 1000 dimensions, more than two ranking SVMs, with small λ values, give rise to nearly perfect ranking results ($\text{MiAP} \approx 1$), showing that the seen tags \mathcal{L} are linearly rank-able under almost every visual association rule—all the ranking constraints imposed by the relevant Y_m and irrelevant $\overline{Y_m}$ tags to image m are satisfied.

However, we shall be cautious before drawing any conclusions *beyond* the experimental vocabulary

\mathcal{L} of seen tags. An image m incurs a visual association rule essentially over all words, though the same rule implies different partitions of distinct experimental vocabularies (e.g., the seen tags \mathcal{L} and unseen ones \mathcal{U}). Accordingly, we would expect the principal direction for the seen tags is also shared by the unseen tags under the same rule, if the answer is YES to the questions at the end of Section 3.2.2.

Generalization to unseen tags. We test whether the same principal direction exists for the seen tags and unseen ones under every visual association rule induced by an image. This can be (only partially) justified by applying the ranking SVMs previously learned, to the unseen tags’ vectors, because we do not know the “true” principal directions. We consider the with 81 unseen tags \mathcal{U} as the “test data” for the trained ranking SVMs, each due to an image incurred visual association. NUS-WIDE provides the annotations of the 81 tags for the images. The results, shown in Figure 3.3(right), are significantly better than the most basic baseline, randomly ranking the tags (the black curve close to the origin), demonstrating that the directions output by SVMs are generalizable to the new vocabulary \mathcal{U} of words.

Observation. Therefore, we conclude that the word vectors are an efficient media to transfer knowledge—the rank-ability along the principal direction—from the seen tags to the unseen ones. We have empirically verified that the visual association rule $(Y_m, \overline{Y_m})$ in words due to an image m can be represented by the linear rank-ability of the corresponding word vectors along a principal direction. Our experiments involve $|\mathcal{L}| + |\mathcal{U}| = 1,006$ words in total. Larger-scale and theoretical studies are required for future work.

3.3 Approximating the linear ranking functions

This section presents our Fast0Tag approach to image tagging. We first describe how to solve image tagging by approximating the principal directions thanks to their existence and generalization, empirically verified in the last section. We then describe detailed approximation techniques.

3.3.1 Image tagging by ranking

Grounded on the observation from Section 3.2, that there exists a principal direction \mathbf{w}_m , in the word vector space, for every visual association rule (Y_m, \bar{Y}_m) in words by an image m , we propose a straightforward solution to image tagging. The main idea is to approximate the principal direction by learning a mapping function $\mathbf{f}(\cdot)$, between the visual space and the word vector space, such that

$$\mathbf{f}(\mathbf{x}_m) \approx \mathbf{w}_m, \quad (3.2)$$

where \mathbf{x}_m is the visual feature representation of the image m . Therefore, given a test image \mathbf{x} , we can immediately suggest a list of tags by ranking the word vectors of the tags along the direction $\mathbf{f}(\mathbf{x})$, namely, by the ranking scores,

$$\langle \mathbf{f}(\mathbf{x}), \mathbf{t} \rangle, \quad \forall \mathbf{t} \in \mathcal{L} \cup \mathcal{U} \quad (3.3)$$

no matter the tags are from the seen set \mathcal{S} or unseen set \mathcal{U} .

We explore both linear and nonlinear neural networks for implementing the approximation function $\mathbf{f}(\mathbf{x}) \approx \mathbf{w}$.

3.3.2 Approximation by linear regression

Here we assume a linear function from the input image representation \mathbf{x} to the output principal direction \mathbf{w} , i.e.,

$$\mathbf{f}(\mathbf{x}) := A\mathbf{x}, \quad (3.4)$$

where A can be solved in a closed form by linear regression. Accordingly, we have the following from the training

$$\mathbf{w}_m = A\mathbf{x}_m + \epsilon_m, m = 1, 2, \dots, M \quad (3.5)$$

where \mathbf{w}_m is the principal direction of all offset vectors of the seen tags, for the visual association rule $(Y_m, \overline{Y_m})$ due to the image m , and ϵ_m are the errors. Minimizing the mean squared errors gives us a closed form solution to A .

One caveat is that we do not know the exact principal directions \mathbf{w}_m at all—the training data only offer images \mathbf{x}_m and the relevant tags Y_m . Here we take the easy alternative and use the directions found by ranking SVMs (cf. Section 3.2) in eq. (3.5). There are thus *two stages* involved to learn the linear function $\mathbf{f}(\mathbf{x}) = A\mathbf{x}$. The first stage trains a ranking SVM over the word vectors of seen tags for each visual association $(Y_m, \overline{Y_m})$. The second stage solves for the mapping matrix A by linear regression, in which the targets are the directions returned by the ranking SVMs.

Discussion. We note that the the linear transformation between visual and word vector spaces has been employed before, e.g., for zero-shot classification [6, 58] and image annotation/classification [197]. This work differs from them with a prominent feature, that the mapped image $\mathbf{f}(\mathbf{x}) = A\mathbf{x}$ has a clear meaning; it depicts the principal direction, which has been empirically verified, for the tags

to be assigned to the image. We next extend the linear transformation to a nonlinear one, through a neural network.

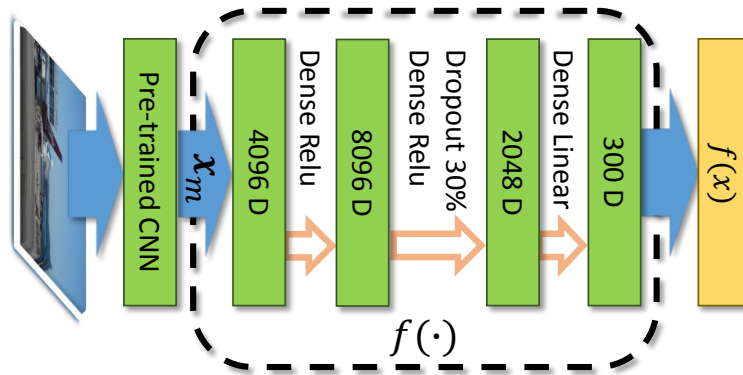


Figure 3.4: The neural network used in our approach for implementing the mapping function $f(x; \theta)$ from the input image, which is represented by the CNN features x , to its corresponding principal direction in the word vector space.

3.3.3 Approximation by neural networks

We also exploit a nonlinear mapping $f(x; \theta)$ by a multi-layer neural network, where θ denotes the network parameters. Figure 3.4 shows the network architecture. It consists of two RELU layers followed by a linear layer to output the approximated principal direction, w , for an input image x . We expect the nonlinear mapping function $f(x; \theta)$ to offer better modeling flexibility than the linear one.

Can we still train the neural network by regressing to the M directions obtained from ranking SVMs? Both our intuition and experiments tell that this is a bad idea. The number M of training instances is small relative to the number of parameters in the network, making it hard to avoid overfitting. Furthermore, the directions by ranking SVMs are not the true principal directions anyway. There is no reason for us to stick to the ranking SVMs for the principal directions.

We instead unify the two stages in Section 3.3.2. Recall that we desire the output of the neural network $\mathbf{f}(\mathbf{x}_m; \boldsymbol{\theta})$ to be the principal direction, along which all the relevant tag vectors $\mathbf{p} \in Y_m$ of an image m rank ahead of the irrelevant ones $\mathbf{n} \in \overline{Y}_m$. Denote by

$$\nu(\mathbf{p}, \mathbf{n}; \boldsymbol{\theta}) = \langle \mathbf{f}(\mathbf{x}_m; \boldsymbol{\theta}), \mathbf{n} \rangle - \langle \mathbf{f}(\mathbf{x}_m; \boldsymbol{\theta}), \mathbf{p} \rangle,$$

the amount of violation to any of those ranking constraints. We minimize the following loss to train the neural network,

$$\begin{aligned} \boldsymbol{\theta}^* &\leftarrow \arg \min_{\boldsymbol{\theta}} \sum_{m=1}^M \omega_m \ell(\mathbf{x}_m, Y_m; \boldsymbol{\theta}), \\ \ell(\mathbf{x}_m, Y_m; \boldsymbol{\theta}) &= \sum_{\mathbf{p} \in Y_m} \sum_{\mathbf{n} \in \overline{Y}_m} \log(1 + \exp\{\nu(\mathbf{p}, \mathbf{n}; \boldsymbol{\theta})\}) \end{aligned} \quad (3.6)$$

where $\omega_m = (|Y_m| |\overline{Y}_m|)^{-1}$ normalizes the per-image RankNet loss [27] $\ell(\mathbf{x}_m, Y_m; \boldsymbol{\theta})$ by the number of ranking constraints imposed by the image m over the tags. This formulation enables the function $\mathbf{f}(\mathbf{x})$ to directly take account of the ranking constraints by relevant \mathbf{p} and irrelevant \mathbf{n} tags. Moreover, it can be optimized with no challenge at all by standard mini-batch gradient descent.

Practical considerations. We use Theano [19] to solve the optimization problem. A mini-batch consists of 1,000 images, each of which incurs on average 4,600 pairwise ranking constraints of the tags—we use all pairwise ranking constraints in the optimization. The normalization ω_m for the per-image ranking loss suppresses the violations from the images with many positive tags. This is desirable since the numbers of relevant tags of the images are unbalanced, ranging from 1 to 20. Without the normalization the MiAP results drop by about 2% in our experiments. For regularization, we use early stopping and a dropout layer [84] with the drop rate of 30%. The optimization hyper-parameters are selected by the validation set.

In addition to the RankNet loss [27] in eq. (3.6), we have also experimented some other choices for the per-image loss, including the hinge loss [41], Crammer-Singer loss [42], and pairwise max-out ranking [94]. The hinge loss performs the worst, likely because it is essentially not designed for ranking problems, though one can still understand it as a point-wise ranking loss. The Crammer-Singer, pairwise max-out, and RankNet are all pair-wise ranking loss functions. They give rise to comparable results and RankNet outperforms the other two by about 2% in terms of MiAP. This may attribute to the ease of control over the optimization process for RankNet. Finally, we note that the list-wise ranking loss [200] can also be employed.

3.4 Experiments on NUS-WIDE

This section presents our experimental results. We contrast our approach to several competitive baselines for the conventional image tagging task on the large-scale NUS-WIDE [39] dataset. Moreover, we also evaluate our method on the zero-shot and seen/unseen image tagging problems (cf. Section 3.2.1). For the comparison on these problems, we extend some existing zero-shot classification algorithms and consider some variations of our own approach.

3.4.1 Dataset and evaluation

3.4.1.1 NUS-WIDE

We mainly use the NUS-WIDE dataset [39] for the experiments in this section. NUS-WIDE is a standard benchmark dataset for image tagging. It contains 269,648 images in the original release and we are able to retrieve 223,821 of them since some images are either corrupted or removed from Flickr. We follow the recommended experiment protocol to split the dataset into a training set with 134,281 images and a test set with 89,603 images. We further randomly separate 20%

from the training set as our validation set for 1) tuning hyper-parameters in our method and the baselines and 2) conducting the empirical analyses in Section 3.2.

3.4.1.2 Annotations of NUS-WIDE

NUS-WIDE releases three sets of tags associated with the images. The first set comprises of 81 “groundtruth” tags. They are carefully chosen to be representative of the Flickr tags, such as containing both general terms (e.g., *animal*) and specific ones (e.g., *dog* and *flower*), corresponding to frequent tags on Flickr, etc. Moreover, they are annotated by high-school and college students and are much less noisy than those directly collected from the Web. This 81-tag set is usually taken as the groundtruth for benchmarking different image tagging methods. The second and the third sets of annotations are both harvested from Flickr. There are 1,000 popular Flickr tags in the second set and nearly 5,000 raw tags in the third.

3.4.1.3 Image features and word vectors

We extract and ℓ_2 normalize the image feature representations of VGG-19 [175]. Both GloVe [148] and Word2vec [124] word vectors are included in our empirical analysis experiments in Section 3.2 and the 300D GloVe vectors are used for the remaining experiments. We also ℓ_2 normalize the word vectors.

3.4.2 Evaluation

We evaluate the tagging results of different methods using two types of metrics. One is the mean image average precision (MiAP), which takes the whole ranking list into account. The other consists of the precision, recall, and F-1 score for the top K tags in the list. We report the results for

$K = 3$ and $K = 5$. Both metrics are commonly used in the previous works on image tagging. We refer the readers to Section 3.3 of [109] for how to calculate MiAP and to Section 4.2 of [75] for the top- K precision and recall.

3.4.3 Conventional image tagging

Here we report the experiments on the **conventional** tagging. The 81 concepts with “groundtruth” annotations in NUS-WIDE are used to benchmark different methods.

3.4.3.1 Baselines

We include TagProp [80] as the first competitive baseline. It is representative among the nearest neighbor based methods, which in general outperform the parametric methods built from generative models [17, 29], and gives rise to state-of-the-art results in the experimental study [109]. We further compare with two most recent parametric methods, WARP [75] and FastTag [34], both of which are built upon deep architectures though using different models. For a fair comparison, we use the same VGG-19 features for all the methods—the code of TagProp and FastTag is provided by the authors and we implement WARP based on our neural network architecture. Finally, we compare to WSABIE [197] and CCA, both correlating images and relevant tags in a low dimensional space. All the hyper-parameters (e.g., the number of nearest neighbors in TagProp and early stopping for WARP) are selected using the validation set.

3.4.3.2 Results

Table 3.1 shows the comparison results of TagProp, WARP, FastTag, WSABIE, CCA, and our Fast0Tag models implemented respectively by the linear mapping and nonlinear neural network.

Table 3.1: Comparison results of the **conventional** image tagging with 81 tags on NUS-WIDE.

Method	%	MiAP	$K = 3$			$K = 5$		
			P	R	F1	P	R	F1
CCA		19	9	15	11	7	20	11
WSABIE [197]		28	16	27	20	12	35	18
TagProp [80]		53	29	50	37	22	62	32
WARP [75]		48	27	45	34	20	57	30
FastTag [34]		41	23	39	29	19	54	28
Fast0Tag (lin.)		52	29	50	37	21	60	31
Fast0Tag (net.)		55	31	52	39	23	65	34

Table 3.2: Comparison results of the **zero-shot** and **seen/unseen** image tagging tasks with 81 unseen tags and 925 seen tags.

Method	%	Zero-shot image tagging						Seen/unseen image tagging							
		MiAP	$K = 3$			$K = 5$			MiAP	$K = 3$			$K = 5$		
			P	R	F1	P	R	F1		P	R	F1	P	R	F1
Random		7.1	2.2	3.8	2.8	2.2	6.1	3.2	1.2	0.6	0.3	0.4	0.6	0.5	0.5
Seen2Unseen		16.7	7.3	12.5	9.2	7.0	19.7	10.3	2.8	2.1	1.1	1.4	1.9	1.6	1.8
LabelEM [7]		23.7	11.9	20.2	14.9	10.2	28.9	15.1	8.8	8.7	4.4	5.8	7.9	6.6	7.2
LabelEM+ [7]		24.9	12.5	21.4	15.8	10.7	30.4	15.8	10.2	11.3	5.7	7.6	9.6	8.1	8.8
ConSE [135]		32.4	17.7	30.1	22.3	13.7	38.8	20.2	12.5	16.7	8.4	11.2	13.5	11.3	12.3
Fast0Tag (lin.)		40.1	21.8	37.2	27.5	17.0	48.4	25.2	18.8	22.9	11.5	15.4	18.7	15.7	17.1
Fast0Tag (net.)		42.2	22.6	38.4	28.4	17.6	50.0	26.0	19.1	21.7	11.0	14.5	18.4	15.5	16.8
RankSVM		37.0	19.7	33.3	24.7	15.2	42.9	22.5	–	–	–	–	–	–	–

We can see that TagProp performs significantly better than WARP and FastTag. However, TagProp’s training and test complexities are very high, being respectively $O(M^2)$ and $O(M)$ w.r.t. the training set size M . In contrast, both WARP and FastTag are more efficient, with $O(M)$ training complexity and constant testing complexity, thanks to their parametric formulation. Our Fast0Tag with linear mapping gives comparable results to TagProp and Fast0Tag with the neural network outperforms the other methods. Also, both implementations have as low computation complexities as WARP and FastTag.

3.4.4 Zero-shot and Seen/Unseen image tagging

This section presents some results for the two novel image tagging scenarios, **zero-shot** and **seen/unseen** tagging.

Fu et al. [61] formalized the **zero-shot** image tagging problem, aiming to annotate test images using a pre-fixed set \mathcal{U} of unseen tags. Our Fast0Tag naturally applies to this scenario, by simply ranking the unseen tags with eq. (3.3). Furthermore, this paper also considers **seen/unseen** image tagging which finds both relevant seen tags from \mathcal{S} and relevant unseen tags from \mathcal{U} for the test images. The set of unseen tags \mathcal{U} could be open and dynamically growing.

In our experiments, we treat the 81 concepts with high-quality user annotations in NUS-WIDE as the unseen set \mathcal{U} for evaluation and comparison. We use the remaining 925 out of the 1000 frequent Flickr tags to form the seen set \mathcal{S} —75 tags are shared by the original 81 and 1,000 tags.

3.4.4.1 Baselines

Our Fast0Tag models can be readily applied to the zero-shot and seen/unseen image tagging scenarios. For comparison we study the following baselines.

Seen2Unseen. We first propose a simple method which extends an arbitrary traditional image tagging method to also working with previously unseen tags. It originates from our analysis experiment in Section 3.2. First, we use any existing method to rank the seen tags for a test image. Second, we train a ranking SVM in the word vector space using the ranking list of the seen tags. Third, we rank unseen (and seen) tags using the learned SVM for zero-shot (and seen/unseen) tagging.

LabelEM. The label embedding method [7] achieves impressive results on zero-shot classification

for fine-grained object recognition. If we consider each tag of $\mathcal{S} \cup \mathcal{U}$ as a unique class, though this implies that some classes will have duplicated images, the LabelEM can be directly applied to the two new tagging scenarios.

LabelEM+. We also modify the objective loss function of LabelEM when we train the model, by carefully removing the terms that involve duplicated images. This slightly improves the performance of LabelEM.

ConSE. Again by considering each tag as a class, we include a recent zero-shot classification method, ConSE [135] in the following experiments.

Note that it is computationally infeasible to compare with [61], which might be the first work to our knowledge on expanding image tagging to handle unseen tags, because it considers all the possible combinations of the unseen tags.

3.4.4.2 Results

Table 3.2 summarizes the results of the baselines and Fast0Tag when they are applied to the zero-shot and seen/unseen image tagging tasks. Overall, Fast0Tag, with either linear or neural network mapping, performs the best.

Additionally, in the table we add two special rows whose results are mainly for reference. The Random row corresponds to the case when we return a random list of tags in \mathcal{U} for zero-shot tagging (and in $\mathcal{U} \cup \mathcal{S}$ for seen/unseen tagging) to each test image. We compare this row with the row of Seen2Unseen, in which we extend TagProp to handle the unseen tags. We can see that the results of Unseen2Seen are significantly better than randomly ranking the tags. This tells us that the simple Seen2Unseen is effective in expanding the labeling space of traditional image tagging methods. Some tag completion methods [173] may also be employed for the same purpose as

Seen2Unseen.

Another special row in Table 3.2 is the last one with RankSVM for zero-shot image tagging. We obtain its results through the following steps. Given a test image, we assume the annotation of the seen tags, \mathcal{S} , are known and then learn a ranking SVM with the default regularization $\lambda = 1$. The learned SVM is then used to rank the unseen tags for this image. One may wonder that the results of this row should thus be the upper bound of our Fast0Tag implemented based on linear regression, because the ranking SVM models are the targets of the linear regressor. However, the results show that they are not. This is not surprising, but rather it reinforces our previous statement that the learned ranking SVMs are not the “true” principal directions. The Fast0Tag implemented by the neural network is an effective alternative for seeking the principal directions.

It would also be interesting to compare the results in Table 3.2 (zero-shot image tagging) with those in Table 3.1 (conventional tagging), because the experiments for the two tables share the same testing images and the same candidate tags; they only differ in which tags are used for training. We can see that the Fast0Tag (net.) results of the zero-shot tagging in Table 3.2 are actually comparable to the conventional tagging results in Table 3.1, particularly about the same as FastTag’s. These results are encouraging, indicating that it is unnecessary to use all the candidate tags for training in order to have high-quality tagging performance.

Table 3.3: Annotating images with up to 4,093 unseen tags.

Method %	MiAP	$K = 3$			$K = 5$		
		P	R	F1	P	R	F1
Random	0.3	0.1	0.1	0.1	0.1	0.1	0.1
Fast0Tag (lin.)	9.8	9.4	7.2	8.2	7.4	9.5	8.4
Fast0Tag (net.)	8.5	8.0	6.2	7.0	6.5	8.3	7.3

3.4.5 Predicting images with 4,093 unseen tags

What happens when we have a large number of unseen tags showing up at the test stage? NUS-WIDE provides noisy annotations for the images with over 5,000 Flickr tags. Excluding the 925 seen tags that are used to train models, there are 4,093 remaining unseen tags. We use the Fast0Tag models to rank all the unseen tags for the test images and the results are shown in Table 3.3. Noting that the noisy annotations weaken the credibility of the evaluation process, the results are reasonably low but significantly higher than the random lists.



Images	Conventional Tagging	Zero-Shot Tagging	See/Unseen Tagging	4k Zero-Shot Tagging	TagProp (Conventional)
	Water Beach Ocean Surf <i>Cat</i>	Water Ocean Surf Beach <i>Snow</i>	Water Ocean Wave Sea Surf	Water Ocean <i>NGO</i> <i>Dam</i> Surf	Water Surf Ocean Beach <i>Whales</i>
	Reflection Water Building Cityscape Harbor	Cityscape Sunset <i>Bridge</i> Reflection Harbor	Night Skyline Cityscape Sunset City	Waterfront <i>Danbe</i> Cityscape Sunset <i>Venice</i>	Water Reflection <i>Sky</i> Building Cityscape
	Coral Fish Ocean Water <i>Rocks</i>	Coral Fish Water Ocean <i>Sand</i>	Coral Underwater Marine Scuba Reef	Coral <i>Korea</i> <i>Mushroom</i> <i>Lichen</i> Fish	Coral Fish Water Ocean <i>Flowers</i>
	Plane Airport Sky Clouds <i>Military</i>	Plane Sky Airport <i>Snow</i> Clouds	Aircraft <i>Aviation</i> Airplane Jet Flying	Aircrafts <i>Takeoff</i> Airline Jets Airlines	Airport Plane Clouds Sky <i>Military</i>
	Train Railroad <i>Bridge</i> Road <i>Fire</i>	Railroad Train <i>Bridge</i> Road <i>Fire</i>	Locomotive Railroad Railway Train Rail	Locomotives Railroad Railways Train Trains	Train Railroad <i>Clouds</i> <i>Sky</i> Road

Figure 3.5: The top five tags for exemplar images [39] returned by Fast0Tag on the conventional, zero-shot, and seen/unseen image tagging tasks, and by TagProp for conventional tagging. (Correct tags: green; mistaken tags: red and italic. Best viewed in color.)

3.4.6 Qualitative results

Figure 3.5 shows the top five tags for some exemplar images [39], returned by Fast0Tag under the conventional, zero-shot, and seen/unseen image tagging scenarios. Those by TagProp under the conventional tagging are shown on the rightmost. The tags in green color appear in the groundtruth annotation; those in red color and *italic* font are the mistaken tags. Interestingly, Fast0Tag performs equally well for traditional and zero-shot tagging and makes even the same mistakes. More results are in Suppl.

3.5 Experiments on IAPRTC-12

We present another set of experiments conducted on the widely used IAPRTC-12 [79] dataset. We use the same tag annotation and image training-test split as described in [80] for our experiments.

There are 291 unique tags and 19627 images in IAPRTC-12. The dataset is split to 17341 training images and 2286 testing images. We further separate 15% from the training images as our validation set.

3.5.1 Configuration

Just like the experiments presented in the last section, we evaluate our methods in three different tasks: **conventional** tagging, **zero-shot** tagging, and **seen/unseen** tagging.

Unlike NUS-WIDE where a relatively small set (81 tags) is considered as the groundtruth annotation, all the 291 tags of IAPRTC-12 are usually used in the previous work to compare different methods. We thus also use all of them conventional tagging.

As for zero-shot and seen/unseen tagging tasks, we exclude 20% from the 291 tags as unseen tags. At the end, we have 233 seen tags and 58 unseen tags.

The visual features, evaluation metrics, word vectors, and baseline methods remain the same as described in the main text.

Table 3.4: Comparison results of the **conventional** image tagging with 291 tags on IAPRTC-12.

Method %	MiAP	$K = 3$			$K = 5$		
		P	R	F1	P	R	F1
TagProp [80]	52	54	29	38	46	41	43
WARP [75]	48	50	27	35	43	38	40
FastTag [34]	48	53	28	36	44	39	41
Fast0Tag (lin.)	46	52	28	37	43	38	40
Fast0Tag (net.)	56	58	31	41	50	44	47

Table 3.5: Comparison results of the **zero-shot** and **seen/unseen** image tagging tasks with 58 unseen tags and 233 seen tags on IAPRTC-12.

Method %	Zero-shot image tagging							Seen/unseen image tagging						
	MiAP	$K = 3$			$K = 5$			MiAP	$K = 3$			$K = 5$		
		P	R	F1	P	R	F1		P	R	F1	P	R	F1
Random	8.1	2.0	4.5	2.8	2.2	2.2	8.1	3.5	2.2	1.2	1.5	1.9	1.7	1.8
Seen2Unseen	15.6	6.1	13.5	8.4	5.3	19.5	8.4	7.2	3.6	1.9	2.5	4.2	3.7	3.9
LabelEM [7]	11.5	3.6	7.9	4.9	3.6	13.3	5.7	13.8	3.1	1.7	2.2	4.4	3.9	8.7
LabelEM+ [7]	17.6	7.3	16.1	10.0	6.4	23.4	10.0	20.1	13.9	7.4	9.7	13.2	11.8	12.5
ConSE [135]	24.1	9.7	21.3	13.3	8.9	32.5	13.9	32.5	38.8	20.6	26.9	31.1	27.6	29.2
Fast0Tag (lin.)	23.1	11.3	24.9	15.6	9.0	33.2	14.2	42.9	50.6	27.0	35.2	40.8	36.2	38.4
Fast0Tag (net.)	20.3	8.5	18.6	11.6	7.2	26.4	11.3	45.9	48.2	25.7	33.5	42.2	37.4	39.7
RankSVM	21.6	10.2	22.6	14.1	8.6	31.7	13.6	–	–	–	–	–	–	–

3.5.2 Results

Table 3.4 and 3.5 show the results of all the three image tagging scenarios (conventional, zero-shot, and seen/unseen tagging). The proposed Fast0Tag still outperforms the other competitive baselines in this new IAPRTC-12 dataset.

A notable phenomenon, which is yet less observable on NUS-WIDE probably due to its noisier seen tags, is that the gap between LabelEM+ and LabelEM is significant. It indicates that the traditional zero-shot classification methods are not suitable for either zero-shot or seen/unseen image tagging task. Whereas we can improve the performance by tweaking LabelEM and by carefully removing the terms in its formulation involving the comparison of identical images.

Images	Conventional Tagging	Zero-Shot Tagging	Seen/Unseen Tagging	4k Zero-Shot Tagging	TagProp (Conventional)
	Lake Mountain Water Sky Reflection	<i>Valley</i> Glacier Mountain Lake Snow	Mountains Valley Glacier Landscape Mountain	<i>Valley</i> Glacier <i>Alpine</i> Mountain Lake	Mountain Snow Lake Water Sky
	Mountain Clouds Snow <i>Sunset</i> Sky	Mountain Snow Glacier Valley Snow	Mountains Mountain Snow Glacier Valley	Mountain <i>Snowy</i> Snow Peaks <i>Alps</i>	Snow Mountain Clouds Glacier Valley
	Harbor Boats Water Ocean Reflection	Boats <i>Sunset</i> <i>Bench</i> Reflection Harbor	<i>Sea</i> Boats Bay Sunset <i>Sailboat</i>	<i>Yachts</i> Waterfront <i>Marina</i> Sailboats <i>Yacht</i>	Boats Water Harbor Ocean <i>Sky</i>
	Water Bridge <i>Castle</i> Sky Reflection	Water Bridge Reflection Boats <i>Cityscape</i>	<i>River</i> Canal Italy Italia <i>Boat</i>	<i>Thames</i> Venice <i>Danube</i> <i>Croatia</i> <i>Quay</i>	Water Reflection Boats Bridge Sky
	Tiger Cat <i>Animal</i> Zebra <i>Tree</i>	<i>Cat</i> Animal Tiger Dog <i>Birds</i>	<i>Zoo</i> Cats Cubs Cat Animal	<i>Meow</i> <i>Paws</i> Cat <i>Cheetah</i> <i>Bengal</i>	Tiger Animal Cat <i>Snow</i> <i>Bear</i>
	Plane Military <i>Airport</i> Sky <i>Fire</i> <i>Clouds</i>	Military Plane Sky <i>Fire</i> <i>Airport</i>	Aircraft Aviation Flying Airplane Flight	<i>USAF</i> <i>Aircrafts</i> USN <i>Squadron</i> <i>Aerobatics</i>	Plane <i>Airport</i> Sky Military <i>Sunset</i>
	Flowers Garden <i>Plants</i> Water leaf	Flowers Garden <i>Grass</i> Leaf <i>Plants</i>	Flowers Flower Green Nature macro	Flowers <i>Wildflowers</i> <i>Blooming</i> Stalk <i>Bouquet</i>	Flowers Garden <i>Plants</i> Water <i>sky</i>

(a)

Images	Conventional Tagging	Zero-Shot Tagging	Seen/Unseen Tagging	TagProp (Conventional)
	Bed Room Wall Lamp Night	Pillow curtain <i>Clock</i> Wood Picture	Pillow Bed Room Wall Curtain	Bed Room Wall Table Wood
	Jersey Cycling Short Cyclist Bike	Racing <i>Frame</i> Helmet <i>Horse</i> <i>Shirt</i>	Cyclist Cycling Bike Jersey Road	Bike Cyclist Short Cycling Jersey
	Child Hand <i>Woman</i> Girl table	Adult <i>Kid</i> <i>Boy</i> <i>Towel</i> Girl	Adult Hand <i>Kid</i> Child <i>Woman</i>	Child Table <i>Tourist</i> Round Hand
	Man Woman House Tree Bench	Park <i>Adult</i> <i>Lion</i> <i>Picture</i> <i>Short</i>	Park Man House Tree Woman	Woman <i>Wall</i> Man Tree <i>People</i>
	Sky City <i>Mountain</i> Building <i>Cloud</i>	Skyline <i>Fountain</i> <i>Cup</i> <i>Boy</i> Shore	Skyline Sky <i>Fountain</i> Building <i>Street</i>	Sky Building City <i>Boat</i> <i>Cloud</i>
	Mountain Snow <i>Cloud</i> <i>Peak</i> Ridge View	Glacier Valley <i>Frame</i> <i>Ridge</i> <i>Skyline</i>	Mountain Snow <i>Cloud</i> <i>Summit</i> Sky	Mountain <i>Cloud</i> Sky Snow <i>Landscape</i>
	Table Woman Plate <i>Man</i> Wall	<i>Bar</i> Shirt Girl <i>Adult</i> <i>Picture</i>	Table Woman Plate <i>Man</i> Wall	Table Woman <i>Man</i> Wall <i>Restaurant</i>

(b)

Figure 3.6: The top five tags for exemplar images in [39](a) and [79](b) returned by Fast0Tag on the conventional, zero-shot, seen/unseen and 4,093 zero-shot image tagging tasks, and by TagProp for conventional tagging. (Correct tags: green; mistaken tags: red and italic)

3.5.3 Qualitative results

In this section, we provide more qualitative results of different tagging methods on both the NUS-WIDE, shown in Figure 3.6.(a) supplementing Figure 3.5, and the IAPRTC-12, shown in Figure 3.6.(b).

Due to incompleteness and noise of tag groundtruth, many actually correct tag predictions are often

evaluated as mistaken predictions since they mismatch with groundtruth. This phenomenon becomes especially apparent in 4k zero-shot tagging results in Figure 3.6.(a) where plentiful diverse tag candidates are considered.

3.6 Summary

We have systematically studied a particular visual regulation over words, the visual association rule which partitions words into two disjoint sets according to their relevances to an image, as well as how it can be captured by the vector offsets in the word vector space. Our empirical results show that, for any image, there exists a principal direction in the word vector space such that the relevant tags' vectors rank ahead of the irrelevant ones' along that direction. The experimental analyses involve 1,006 words; larger-scale and theoretical analyses are required for future work. Built upon this observation, we develop a Fast0Tag model to solve image tagging by estimating the principal directions for input images. Our approach is as efficient as FastTag [34] and is capable of annotating images with a large number of previously **unseen** tags. Extensive experiments validate the effectiveness of our Fast0Tag approach.

CHAPTER 4: DOMAIN ADAPTATION FOR SEMANTIC SEGMENTATION

4.1 Problem Introduction

Semantic Segmentation segmentation is one of the most challenging and fundamental problems in computer vision. It assigns a semantic label to each pixel of an input image [67]. The resulting output is a dense and rich annotation of the image, with one semantic label per pixel. Semantic segmentation facilitates many downstream applications, including autonomous driving, which, over the past few years, has made great strides towards use by the general population. Indeed, several datasets and test suites have been developed for research on autonomous driving [40, 202, 170, 47, 156], and, among them, semantic segmentation is often considered one of the key tasks.

Convolutional neural networks (CNNs) [114, 99] have become a hallmark backbone model to solve the semantic segmentation of large-scale image sets over the last half decade. All of the top-performing methods on the challenge board of the Cityscapes pixel-level semantic labeling task [40] rely on CNNs. One of the reasons that CNNs are able to achieve a high level of accuracy for this task is that the training set is sufficiently large and well-labeled, covering the variability of the test set for the research purpose. In practice, however, it is often hard to acquire new training

This chapter contains previously published materials from: 1) “Curriculum domain adaptation for semantic segmentation of urban scenes” by Yang Zhang, Philip David, and Boqing Gong, published in Proceedings of the IEEE International Conference on Computer Vision in 2017; 2) “A Curriculum Domain Adaptation Approach to the Semantic Segmentation of Urban Scenes” by Yang Zhang, Philip David, Hassan Foroosh, Boqing Gong, published in IEEE Transactions on Pattern Analysis and Machine Intelligence in 2019.

sets that fully cover the huge variability of real-life test scenarios. Even if one could compose a large-scale dataset with sufficient variability, it would be extremely tedious to label the images with pixel-wise semantic labels. For example, Cordts et al. report that the annotation and quality control took more than 1.5 hours per image in the popular Cityscapes dataset [40].

These challenges motivate researchers to approach the segmentation problem by using complementary synthetic data. With modern graphics engines, automatically synthesizing diverse urban-scene images along with pixel-wise labels would require very little to zero human labor. Figure 4.4 shows some synthetic images of the GTA dataset [156]. They are quite photo-realistic, giving rise to the hope that a semantic segmentation neural network trained from them can perform reasonably well on the real images as well. However, our experiments show that this is not the case (cf. Section 4.3), signifying a severe mismatch between the real images and the synthesized ones. Multiple factors may contribute to the mismatch, such as the scene layouts, capture devices (camera vs. rendering engine), view angles, lighting conditions and shadows, textures, etc.

In this chapter, our main objective is to investigate the use of domain adaptation techniques [85, 86, 210, 147] to more effectively transfer the semantic segmentation neural networks trained using synthetic images to high-quality segmentation networks for real images. We build our efforts upon our prior work [210], in which we propose a novel domain adaptation approach to the semantic segmentation of urban scenes.

Domain adaption, which mainly aims to boost models' performance when the target domain of interest differs from the one where the models are trained, has long been a popular topic in machine learning and computer vision [43]. It has recently drawn even greater attention along with transfer learning [195] thanks to the prevalence of deep neural networks which are often "data-hungry". An intuitive domain adaptation strategy is to learn domain-invariant feature representations for the images of both domains, where the source domain supplies a labeled training set and the target

domain reveals zero to a few labeled images along with many unlabeled ones. In this case, the source domain features would resemble the target ones’ characteristics. Thus, the model trained on the labeled source domain can be generalized to the target domain. Earlier “shallow” methods achieve such goals by exploiting various intrinsic structures of the data [74, 76, 55, 178, 137, 72, 10, 100]. In contrast, the recent “deep” methods mainly devise new loss functions and/or network architectures to add domain-invariant ingredients to the gradients backpropagating through the neural networks [191, 115, 190, 66, 65].

Upon observing the success of learning domain-invariant features in the prior domain adaptation tasks, it is a natural tendency to follow the same principle for the adaptation of semantic segmentation models. There have been some positive results along this line [86, 147]. However, the underlying assumption of this principle may prevent the methods designed around it from achieving high adaptation performance. By focusing on learning domain-invariant features X (i.e., such that $P_S(X) \approx P_T(X)$, where the subscripts S and T stand for the source and target domains, respectively), one assumes the conditional distribution $P(Y|X)$, where Y are the pixel labels, is more or less shared by the two domains. This assumption is less likely to be true when the classification boundary becomes more and more sophisticated — the prediction function for semantic segmentation has to be sophisticated. The sets of pixel labels are high-dimensional, highly structured, and interdependent, implying that the learner has to resolve the predictions in an exponentially large label space. Besides, some discriminative cues in the data would be suppressed if one matches the feature representations of the two domains without taking careful account of the structured labels. Finally, data instances are the proxy to measure the domain difference [78, 65, 66]. However, it is not immediately clear what comprises the instance in semantic segmentation [86], especially given that the top-performing segmentation methods are built upon deep neural networks [114, 144, 134, 32]. Hoffman et al. take each spatial unit in the fully convolutional network (FCN) [114] as an instance [86]. We contend that such instances are actually non-i.i.d. in either

individual domain, as their receptive fields overlap with each other.

How can we avoid the assumption that the source and target domains share the same prediction function in a transformed domain-invariant feature space? Our proposed solution draws on two key observations. One is that the urban traffic scene images have strong idiosyncrasies (e.g., the size and spatial relations of buildings, streets, cars, etc.). Therefore, *some tasks are “easy” and, more importantly, suffer less because of the domain discrepancy*. For instance, it is easy to infer from a traffic scene image that the road often occupies a larger number of pixels than the traffic sign does. Second, the structured output in semantic segmentation enables convenient posterior regularization [64], as opposed to the generic (e.g., ℓ_2) regularization over model parameters.

Accordingly, we propose a curriculum-style [18] domain adaptation approach. Recall that, in domain adaptation, only the source domain supplies many labeled data while there are no or only scarce labels from the target domain. Our curriculum domain adaptation begins with the easy tasks, in order to gain some high-level properties about the unknown pixel-level labels for each target image. It then learns a semantic segmentation network, the hard task, whose predictions over the target images are constrained to follow those target-domain properties as much as possible.

To develop the easy tasks for the curriculum, we consider estimating label distributions over both global images and some landmark superpixels of the target domain. Take the former for instance. The label distribution indicates the percentage of pixels in an image that correspond to each category. We argue that these tasks are easier, despite the domain mismatch, than predicting pixel-wise labels. The label distributions are only rough estimations about the labels’s statistics. Moreover, the size relations between road, building, sky, people, etc. constrain the shape of the distributions, effectively reducing the search space. Finally, models to estimate the label distributions over superpixels may benefit from the urban scenes’ canonical layout that transcends domains, e.g., buildings stand beside streets.

Why and when are these seemingly simple label distributions useful for the domain adaptation of semantic segmentation? In our experiments, we find that the segmentation networks trained on the source domain perform poorly on many target images, giving rise to disproportionate label assignments (e.g., many more pixels are classified to sidewalks than to streets). To rectify this, the image-level label distribution informs the segmentation network *how* to update the predictions while the label distributions of the anchor superpixels tell the network *where* to update. Jointly, they guide the adaptation of the networks to the target domain to, at least, generate proportional label predictions. Note that additional “easy tasks” can be incorporated into our approach in the future.

Our main contribution is the proposed curriculum-style domain adaptation for the semantic segmentation of urban scenes. We select for the curriculum the easy and useful tasks of inferring label distributions for both target images and landmark superpixels in order to gain some necessary properties about the target domain. Built upon these, we learn a pixel-wise discriminative segmentation network from the labeled source data and, meanwhile, conduct a “sanity check” to ensure the network behavior is consistent with the previously learned knowledge about the target domain. Our approach effectively eludes the assumption about the existence of a common prediction function for both domains in a transformed feature space. It readily applies to different segmentation networks as it does not change the network architecture or impact any intermediate layers.

Beyond our prior work [210], we provide more algorithmic details and experimental studies about our approach, including new experiments using the GTA dataset [156] and ablation studies about the number of superpixels, feature representations of the superpixels, various backbone neural networks, prediction confusion matrices, etc. In addition, we introduce a color constancy scheme into our framework, which significantly improves the adaptation performance and may be plugged into any domain adaptation method as a standalone image pre-processing step. We also quantitatively

measure the “market value” of the synthetic data to reveal how much cost it could save out of the labeling of real images. Finally, we provide a comprehensive survey of domain adaptation for semantic segmentation works that were published after ours [210]. We group them into different categories and experimentally demonstrate that other methods are complementary to ours.

4.2 Approach

In this section, we present the details of our approach to curriculum domain adaptation for the semantic segmentation of urban scene images. Unlike previous works [143, 86] that align the domains via an intermediate feature space and thereby implicitly assume the existence of a single decision function for the two domains, it is our intuition that, for structured prediction (i.e., semantic segmentation here), the cross-domain generalization of machine learning models can be more efficiently improved if we avoid this assumption and instead train them subject to necessary properties they should retain in the target domain. After a brief introduction on the preliminaries, we describe in Section 4.2.2 how to facilitate semantic segmentation adaptation during training using estimated target domain properties. Then, in Section 4.2.3, we focus on the types of target domain properties and how to estimate them.

4.2.1 Preliminaries

In particular, the properties of interest concern pixel-wise category labels $Y_t \in \mathbb{R}^{W \times H \times C}$ of an arbitrary image $I_t \in \mathbb{R}^{W \times H}$ from the target domain, where W and H are the width and height of the image, respectively, and C is the number of categories. We use a one-hot vector encoding for the groundtruth labels, i.e., $Y_t(i, j, c)$ takes the value of either 0 or 1, where the latter means that the c -th label is assigned by a human annotator to the pixel at (i, j) . Correspondingly, the prediction

$\widehat{Y}_t(i, j, c) \in [0, 1]$ by a segmentation network is realized by a softmax function per pixel.

We express each target property in the form of a distribution p_t over the C categories, where $\sum_{c=1}^C p_t(c) = 1$ and $0 \leq p_t(c), \forall c$. $p_t(c)$ represents the occupancy proportion of the category c over the t -th target image or a superpixel of that image. Therefore, one can immediately calculate the distribution p_t given the human annotations Y_t to the image. For instance, the image-level label distribution is expressed by

$$p_t(c) = \frac{1}{WH} \sum_{i=1}^W \sum_{j=1}^H Y_t(i, j, c), \quad \forall c. \quad (4.1)$$

Similarly, we can compute the estimated target property/distribution from the network predictions \widehat{Y}_t and denote it by \widehat{p}_t ,

$$\widehat{p}_t(c) = \frac{1}{WH} \sum_{i=1}^W \sum_{j=1}^H \left(\frac{\widehat{Y}_t(i, j, c)}{\max_{c'}(\widehat{Y}_t(i, j, c'))} \right)^K, \quad \forall c \quad (4.2)$$

where $K > 1$ is a large constant whose effect is to “sharpen” the softmax activation per pixel $\widehat{Y}_t(i, j, c)$ such that the summand is either 1 or very close to 0, in a similar shape as the summand $Y_t(i, j, c)$ of eq. (4.1). We set $K = 6$ in our experiment as larger K caused numerical instability. Finally, we ℓ_1 -normalize the vector $(\widehat{p}_t(1), \widehat{p}_t(2), \dots, \widehat{p}_t(C))^T$ such that its elements are all greater than 0 and sum up to 1 — in other words, the vector remains a valid discrete probability distribution.

4.2.2 Domain adaptation observing the target properties

Ideally, we would like to have a segmentation network to imitate human annotators of the target domain. Therefore, necessarily, the properties of their annotation results should be the same too.

We capture this notion by minimizing the cross entropy $\mathcal{C}(p_t, \hat{p}_t) = H(p_t) + \text{KL}(p_t, \hat{p}_t)$ at training, where the first term of the right-hand side is the entropy and the second is the KL-divergence.

Given a mini-batch consisting of both source images (S) and target images (T), the overall objective function for training the cross-domain generalizing segmentation network is

$$\min \frac{\gamma}{|S|} \sum_{s \in S} \mathcal{L}(Y_s, \hat{Y}_s) + \frac{1-\gamma}{|T|} \sum_{t \in T} \sum_k \mathcal{C}(p_t^k, \hat{p}_t^k) \quad (4.3)$$

where \mathcal{L} is the pixel-wise cross-entropy loss defined over the fully labeled source domain images, enforcing the network to have the pixel-level discriminative capabilities, and the second term is over the unlabeled target domain images, hinting the network what necessary properties its predictions should have in the target domain. We use $\gamma \in [0, 1]$ to balance the two strengths in training and superscript k to index different types of label distributions (cf. p_t in eq. (4.1) and Section 4.2.3).

Note that, in the unsupervised domain adaptation context, we actually cannot directly compute the label distributions $\{p_t^k\}$ because the groundtruth annotations of the target domain are unknown. Nonetheless, using the labeled source domain data, these distributions are easier to estimate than are the labels for every pixel of a target image. We present two types of such properties and the details for inferring them in the next section. In future work, it is worth exploring other properties.

Remarks. Mathematically, the objective function has a similar form as model compression [26, 83]. Hence, we borrow some concepts to gain a more intuitive understanding of our domain adaptation procedure. The “student” network follows a curriculum to learn simple knowledge about the target domain before it addresses the hard one of semantically segmenting images. The models inferring the target properties act like “teachers”, as they hint what label distributions the final solution (image annotation) may have in the target domain at the image and superpixel levels.

Another perspective is to understand the target properties as a posterior regularization [64] for

the network. The posterior regularization can conveniently encode a priori knowledge into the objective function. Some applications using this approach include weakly supervised segmentation [144, 160] and detection [20] and rule-regularized training of neural networks [89]. In addition to the domain adaptation setting and novel target properties, another key distinction of our work is that we decouple the label distributions from the network predictions and thus avoid the EM type of optimization, which is often involved and incurs extra computational overhead. Our approach learns the segmentation network with almost effortless changes to the popular deep learning tools.

4.2.3 *Inferring the target properties*

Thus far, we have presented the “hard” task — learning the segmentation neural network — in the curriculum domain adaptation. In this section, we describe the “easy” tasks, i.e., how to infer the target domain properties without any annotations from the target domain. Our contributions also include selecting the particular form of label distributions to constitute the simple tasks.

4.2.3.1 *Global label distributions over images*

Due to the domain disparity, a baseline segmentation network trained on the source domain (i.e., using the first term of eq. (4.3)) could be easily crippled given the target images. In our experiments, we find that our baseline network constantly mistakes streets for sidewalks and/or cars (cf. Figure 4.3). Consequently, the predicted labels for the pixels are highly disproportionate.

To rectify this, we employ the label distribution p_t over the global image as our first property (cf. eq. (4.1)). Without access to the target labels, we have to train machine learning models from the labeled source images to estimate the label distribution p_t of a target image. Nonetheless, we argue that this is less challenging than generating per-pixel predictions despite that both tasks are

influenced by the domain mismatch.

In our experiments, we examine several different approaches to this task. We extract 1536D image features from the output of the average pooling layer in Inception-Resnet-v2 [180] as the input to the following models.

Logistic regression. Although multinomial logistic regression (LR) is mainly used for classification, its output is actually a valid distribution over the categories. For our purpose, we thus train it by replacing the one-hot vectors in the cross-entropy loss with the groundtruth label distribution p_s , which is counted by using eq. (4.1) from the human labels of the source domain. Given a target image, we directly take the LR’s output as the estimated label distribution p_t .

Mean of nearest neighbors. We also test a nonparametric method by simply retrieving multiple nearest neighbor (NN) source images for each target image and then transferring the mean of their label distributions to the target image. We use the ℓ_2 distance in the Inception-Resnet-v2 feature space for the NN retrieval.

Finally, we include two dumb predictions as the control experiments. One is, for any target image, to output the mean of all the label distributions in the source domain (**source mean**), and the other is to output a **uniform distribution**.

4.2.3.2 *Local label distributions of landmark superpixels*

The image-level label distribution globally penalizes potentially disproportional segmentation output in the target domain. It is yet inadequate in providing spatial regularization to the network. In this section, we consider the use of label distributions over some superpixels as the anchors to

drive the network towards spatially desired target properties.

Note that it is not necessary, and is even harmful, to use all of the superpixels in a target image to regularize the segmentation network because it would be too strong a force and might overrule the pixel-wise discriminativeness (obtained from the fully labeled source domain), especially when the label distributions are not inferred accurately enough.

In order to have the dual effect of both estimating the label distributions of some superpixels and selecting them from all candidate superpixels, we employ a linear SVM in this work. We first segment each image into 100 superpixels using linear spectral clustering [110]. For the superpixels of the source domain, we are able to assign a single dominant label to each of them and then train a multi-class SVM using the “labeled” superpixels of the source domain. Given a test superpixel of a target image, the multi-class SVM returns a class label as well as a decision value, which is interpreted as the confidence score about classifying this superpixel. We keep the top 30% most confident superpixels in the target domain. The class labels are then encoded into one-hot vectors which serve as valid distributions about the category labels upon the selected landmark superpixels. Albeit simple, we find this method works very well.

In order to train the aforementioned superpixel SVM, we need to find a way to represent the superpixels in a feature space. We encode both visual and contextual information to represent a superpixel. First, we use the FCN-8s [114] pre-trained on the PASCAL CONTEXT [127] dataset, which has 59 distinct classes, to obtain 59 detection scores for each pixel. We then average these scores within each superpixel. The final feature representation of a superpixel is a 295D concatenation of the 59D vectors of itself, its left and right superpixels, as well as the two respectively above and below it. As this feature representation relies on extra data source, we also examine handcrafted features and VGG features [175] in the experiments.

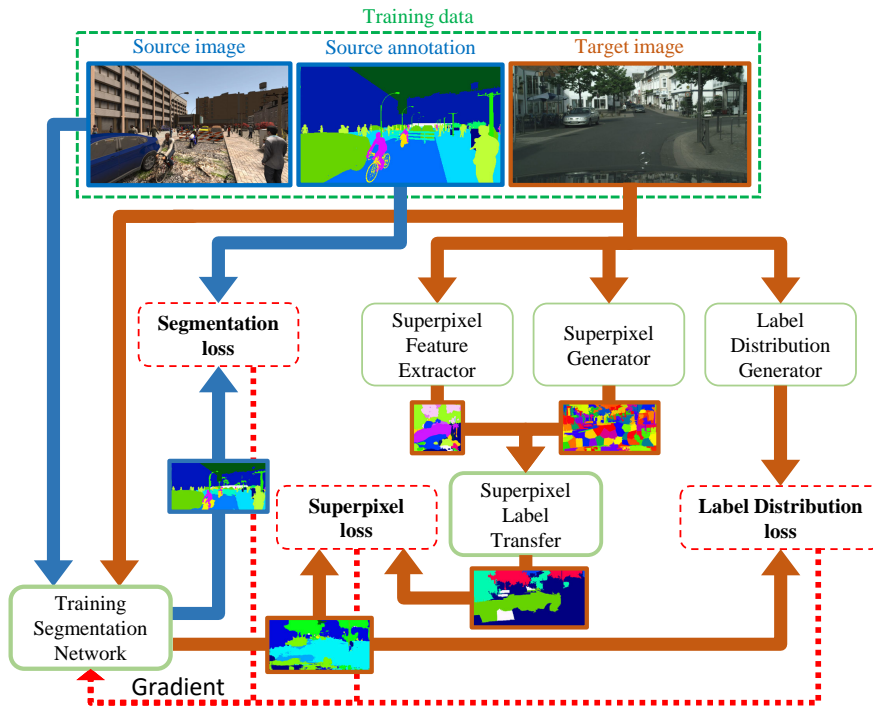


Figure 4.1: The overall framework of our curriculum domain adaptation approach to the semantic segmentation of urban scenes.

4.2.4 Curriculum domain adaptation: recapitulation

We recap the proposed curriculum domain adaptation using Figure 4.1 before presenting the experiments in the next section. Our main idea is to execute the domain adaptation step by step, starting from the easy tasks that, compared to semantic segmentation, are less sensitive to domain discrepancy. We choose the label distributions over global images and local landmark superpixels in this work; more tasks will be explored in the future. The solutions to them provide useful gradients originating from the target domain (cf. the arrows with brown color in Figure 4.1), while the source domain feeds the network with well-labeled images and segmentation masks (cf. the dark blue arrows in Figure 4.1).

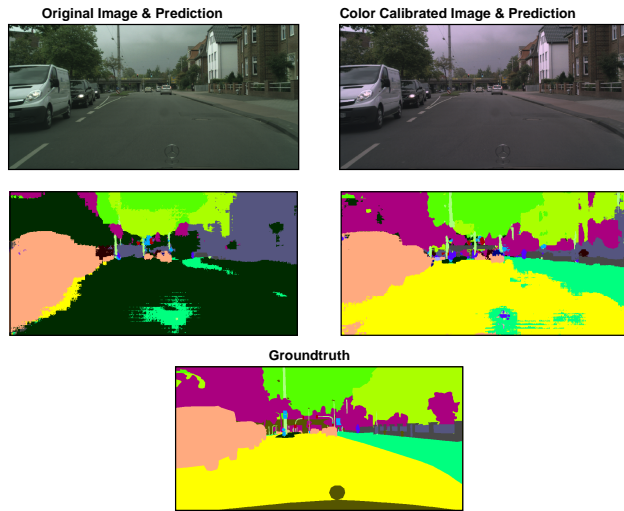


Figure 4.2: Predictions by the same FCN-8s model, without domain adaptation, before and after we calibrate the image’s colors.

4.2.5 Color Constancy

In this subsection, we propose a color calibration preprocessing step which we find very effective in adapting semantic segmentation networks from the source synthetic domain to the target real image domain. We assume the colors of the two domains are drawn from different distributions. We calibrate the target domain images’ colors to those of the source domain, hence reducing their discrepancy in terms of the color. We describe it here as an independent subsection because it can stand alone and can be added to any existing methods for the domain adaptation of semantic segmentation.

Humans have the ability to perceive the same color of an object even when it is exposed in different illuminations [56], but image capturing sensors do not. As a result, different illuminations result in distinct RGB images captured by the cameras. Consequently, the perception incoherence hinders the performance of computer vision algorithms [16] because the illumination is often among the key factors that cause the domain mismatch. To this end, we propose to use computational color

constancy [71] to eliminate the influence of illuminations.

The goal of color constancy is to correct the colors of images acquired under unconventional or biased lighting sources to the colors that are supposed to be under the reference lighting condition. In our domain adaptation scenario, we assume that the source domain resembles the reference lighting condition. We learn a parametric model to describe both the target and source lighting sources and then try to restore the target images according to the source domain’s light. However, not all the color constancy methods are applicable. For instance, some methods rely on physics priors or the statistics of natural images, both of which are unavailable in synthetic images.

Due to the above concerns, we instead use a gamut-based color constancy method [70] to align the target and source images in terms of their colors. This method infers the property of the light source under the assumption that only a limited range of color could be observed under a certain light source. This matches our assumption that the target images’ colors and the source images’ colors belong to different distributions/ranges. In addition to the pixel values, the image edge and derivatives are also used to find the mapping. While we omit the details of this color constancy method and refer the readers to [70] instead, we show in Figure 4.2 how sensitive the segmentation model is to the illumination. We can see that, prior to applying color constancy, a large part of a CityScapes image is incorrectly classified as “terrain” only because the image is a little greenish.

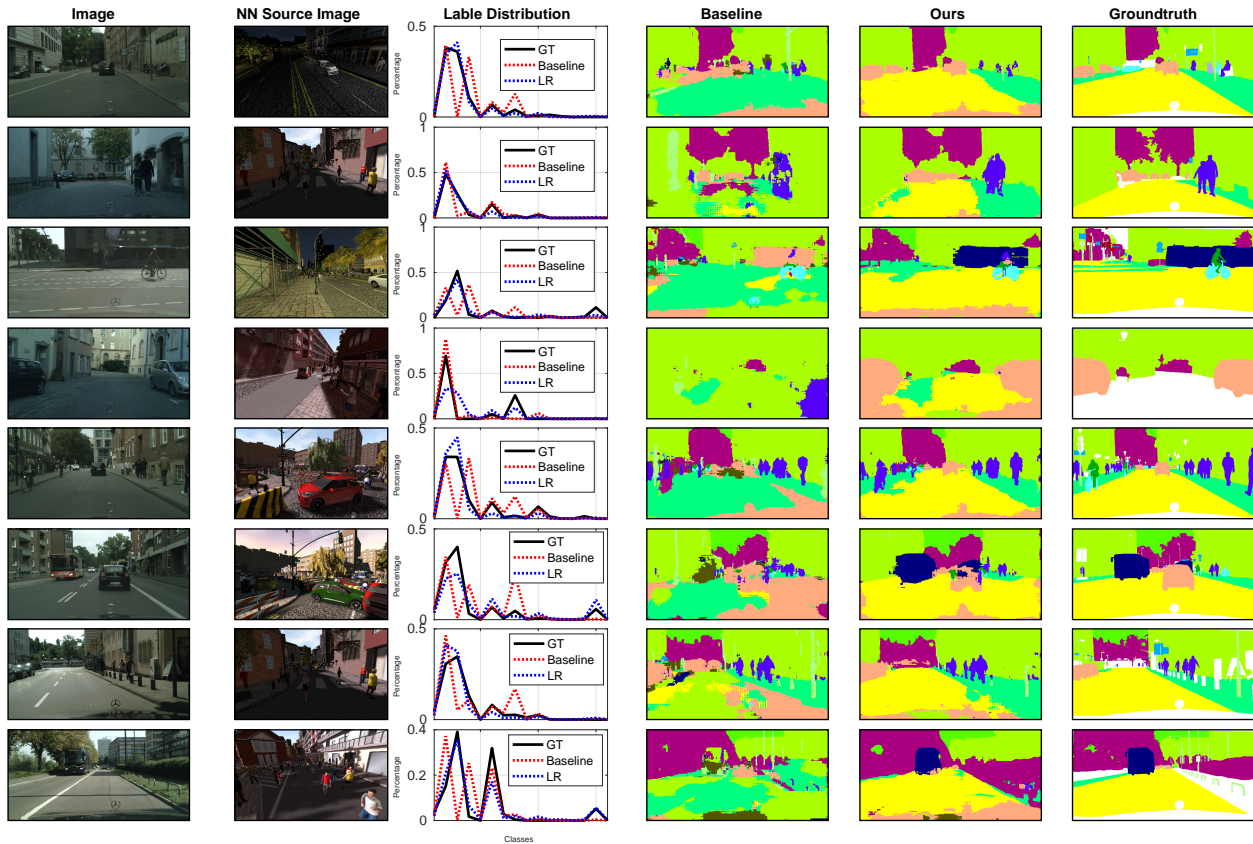


Figure 4.3: Qualitative semantic segmentation results on the Cityscapes dataset [158] (target domain). For each target image in the first column, we retrieve its nearest neighbor from the SYNTHIA [40] dataset (source domain). The third column plots the label distributions due to the groundtruth pixel-wise semantic annotation, the predictions by the baseline network with no adaptation, and the inferred distribution by logistic regression. The last three columns are the segmentation results by the baseline network, our domain adaptation approach, and human annotators, respectively.

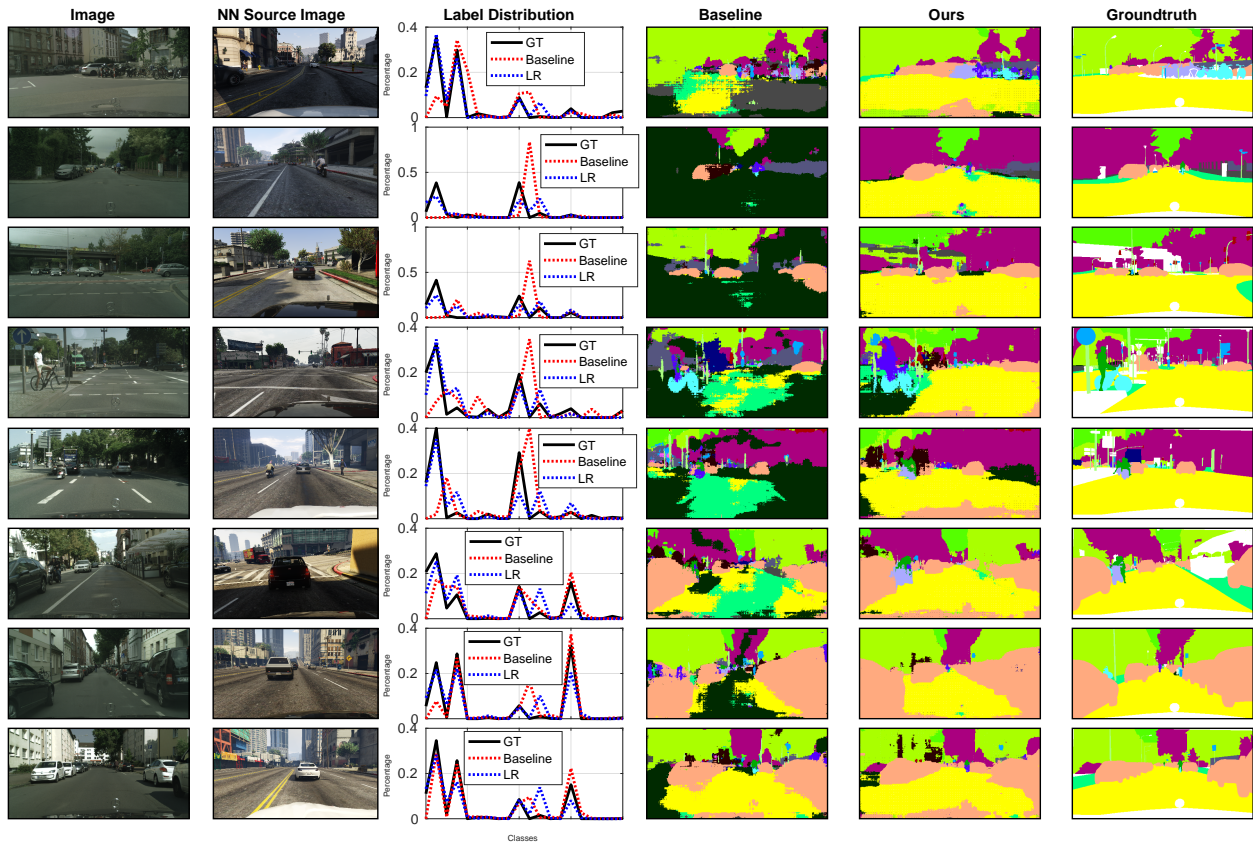


Figure 4.4: Qualitative semantic segmentation results on the Cityscapes dataset [158] (target domain). For each target image in the first column, we retrieve its nearest neighbor from the GTA [156] dataset (source domain). The third column plots the label distributions due to the groundtruth pixel-wise semantic annotation, the predictions by the baseline network with no adaptation, and the inferred distribution by logistic regression. The last three columns are the segmentation results by the baseline network, our domain adaptation approach, and human annotators, respectively.

4.3 Experiments

In this section, we run extensive experiments to verify the effectiveness of our approach and study several variations of it to gain a thorough understanding about how different components contribute to the overall results. We investigate the global image-wise label distribution and the local

landmark superpixels by separate experiments. We also empirically examine the effects of distinct granularities of the superpixels as well as different feature representations of the superpixels. Moreover, we compare, on two synthetic datasets, our approach to several competing baselines for the adaptation of deep neural networks in urban scene segmentation. We use confusion matrices to show how different classes of the urban scene images intervene with each other. Finally, we run few-shot adaptation experiments by gradually revealing some labels of the images of the target domain. The results show that, even when more than 1,000 target images are labeled (50% of the Cityscapes training set [40]), the adaptation from synthetic images is still able to boost the results by a relatively large margin.

4.3.1 Segmentation network and optimization

In most of our experiments, we use FCN-8s [114] as our semantic segmentation network. We initialize its convolutional layers with VGG-19 [175] and then train it using the AdaDelta optimizer [207] with default parameters. Each mini-batch contains five source images and five randomly chosen target images. When we train the baseline network with no adaptation, however, we try to use the largest possible mini-batch which includes 15 source images. The network is implemented in Keras [57] and Theano [9]. We train different versions of the network on a single Tesla K40 GPU.

Additionally, we run an ablation experiment using the state-of-the-art segmentation neural network ADEMAPP [199]. The training details and network architecture are described in Section 4.3.7.

We note that our curriculum domain adaptation can be readily applied to other segmentation networks (e.g., [134, 32]). Once we infer the label distributions of the unlabeled target images and some of their landmark superpixels, we can use them to train different segmentation networks by eq. (4.3) without changing them.

4.3.2 Datasets and evaluation

We use the publicly available **Cityscapes** [40] as our target domain in the experiments. For the source domains of synthesized images, we test both **GTA** [156] and **SYNTHIA** [158].

Cityscapes is a real-world vehicle-egocentric image dataset collected from 50 cities in Germany and nearby countries. It provides four disjoint subsets: 2,993 training images, 503 validation images, 1,531 test images, and 20,021 auxiliary images. All the training, validation, and test images are accurately annotated with per-pixel category labels, and the auxiliary set is coarsely labeled. There are 34 distinct categories in the dataset. Among them, 19 categories are officially recommended for training and evaluation.

SYNTHIA [158] is a large dataset of synthetic images and provides a particular subset, called SYNTHIA-RAND-CITYSCAPES, to pair with Cityscapes. This subset contains 9,400 images that are automatically annotated with 12 object categories, one void class, and some unnamed classes. Note that the virtual city used to generate the synthetic images does not correspond to any of the real cities covered by Cityscapes.

GTA [156] is a synthetic vehicle-egocentric image dataset collected from the open world in a realistically rendered computer game, Grand Theft Auto V (GTA). It contains 24,996 images. Unlike the SYNTHIA dataset, its semantic segmentation annotation is fully compatible with the Cityscapes dataset. Hence, we use all of the 19 official training classes in our experiments.

4.3.2.1 Domain idiosyncrasies

Although all datasets depict urban scenes and both SYNTHIA and GTA are created to be as photo-realistic as possible, they are mismatched domains in several ways. The most noticeable difference

is probably the coarse-grained textures in SYNTHIA; very similar texture patterns repeat in a regular manner across different images. The textures in GTA are better but still visibly artificial. In contrast, the Cityscapes images are captured by high-quality dash-cameras. Another major distinction is the variability in view angles. Since Cityscapes images are recorded by the dash cameras mounted on a moving car, they are viewed from almost a constant angle that is about parallel to the ground. More diverse view angles are employed by SYNTHIA — it seems like some cameras are placed on the buildings that are significantly higher than a bus. Most GTA images are dashcam images, but some are captured from the view points of pedestrians. In addition, some of the SYNTHIA images are severely shadowed by extreme lighting conditions, while we find no such conditions in the Cityscapes images. Finally, there is a subtle difference in color between the synthetic images and the real ones due to the graphics rendering engines’ systematic performance. For instance, the GTA images are overly saturated (cf. Figure 4.4) and SYNTHIA images are overly bright in general. These combined factors, among others, make the domain adaptation from SYNTHIA and GTA to Cityscapes a very challenging problem.

Figure 4.3 and Figure 4.4 show some example images from the three datasets. We pair each Cityscapes image with its nearest neighbor in SYNTHIA/GTA, retrieved by the Inception-Resnet-v2 [180] features. However, many of the cross-dataset nearest neighbors are visually very different from the query images, verifying the dramatic disparity between the two domains.

4.3.2.2 Evaluation

We use the evaluation code released along with the Cityscapes dataset to evaluate our results. It calculates the PASCAL VOC intersection-over-union, i.e., $\text{IoU} = \frac{\text{TP}}{\text{TP} + \text{FP} + \text{FN}}$ [48], where TP, FP, and FN are the numbers of true positive, false positive, and false negative pixels, respectively, determined over the whole test set. Since we have to resize the images before feeding them to

the segmentation network, we resize the output segmentation mask back to the original image size before running the evaluation against the groundtruth annotations.

Table 4.1: χ^2 distances between the groundtruth label distributions and those predicted by different methods for the adaptation from SYNTHIA to Cityscapes.

Method	Uniform	NoAdapt	Src mean	NN	LR
χ^2 Distance	1.13	0.65	0.44	0.33	0.27

4.3.3 Results of inferring global label distribution

Before presenting the final semantic segmentation results, we first compare different approaches to inferring the global label distributions over the target images (cf. Section 4.2.3.1). We use SYNTHIA and Cityscapes’ held-out validation images as the source domain and the target domain, respectively, in this experiment.

In Table 4.1, we compare the estimated label distributions with the groundtruth ones using the χ^2 distance, the smaller the better. We see that the baseline network (NoAdapt), which is directly learned from the source domain without any adaptation methods, outperforms the dumb uniform distribution (Uniform) and yet no other methods. This confirms that the baseline network gives rise to severely disproportional predictions on the target domain.

Another dumb prediction (Src mean), i.e., using the mean of all label distributions over the source domain as the prediction for any target image, however, performs reasonably well mainly because the protocol layouts of the urban scene images. This result implies that the images from the simulation environments share at least similar layouts as the real images, indicating the potential value of the simulated source domains for the semantic segmentation task of urban scenes.

Finally, the nearest neighbor (NN) based method and the multinomial logistic regression (LR) (cf. Section 4.2.3.1) perform the best. We use the output of LR on the target domain in our remaining experiments.

4.3.4 Domain adaptation experiments

In this section, we present our main results of this chapter, i.e., comparison results for the domain adaptation from simulation to real images for the semantic segmentation task. Here we focus on the base segmentation neural network FCN-8s [114]. Another network, ADEMAPP [199], is studied in Section 4.3.7.

Since our ultimate goal is to solve the semantic segmentation problem for real images of urban scenes, we take Cityscapes as the target domain and SYNTHIA/GTA as the source domain. We split 500 images out of the Cityscapes training set for validation (i.e., to monitor the convergence of the networks). In training, we randomly sample mini-matches from both the images and labels of SYNTHIA/GTA and the remaining images of Cityscapes yet with no labels. The original Cityscapes validation set is used as our test set.

All of the 19 classes provided by GTA are used in the experiments. For the adaptation from SYNTHIA to Cityscapes, we manually find 16 common classes between the two datasets: sky, building, road, sidewalk, fence, vegetation, pole, car, traffic sign, person, bicycle, motorcycle, traffic light, bus, wall, and rider. The last four are unnamed and yet labeled in SYNTHIA.

4.3.4.1 Baselines

We mainly compare our approach to the following competing methods. Section 4.4 supplies additional discussions about and comparisons with more related works.

No adaptation (NoAdapt). We directly train the FCN-8s model on the source domain (SYNTHIA or GTA) without applying any domain adaptation methods. This is the most basic baseline in our experiments.

Superpixel classification (SP). Recall that we have trained a multi-class SVM using the dominant labels of the superpixels in the source domain. We then use it to classify the target superpixels.

Landmark superpixels (SP Lndmk). We keep the top 30% most confidently classified superpixels as the landmarks to regularize our segmentation network during training (cf. Section 4.2.3.2). It is worth examining the classification results of these superpixels. We execute the evaluation after assigning the void class label to the other pixels of the images. In addition to the IoU, we have also evaluated the classification results of the superpixels by accuracy for the domain adaptation experiments from SYNTHIA to Cityscapes. We find that the classification accuracy is 71% for all superpixels of the target domain. For the top 30% landmark superpixels, the classification accuracy is more than 88%.

FCNs in the wild (FCN Wld). Hoffman et al.’s work [86] was, to the best of our knowledge, the only existing one addressing the same problem as ours when we published the conference version [210] of this work. They introduce a pixel-level adversarial loss to the intermediate layers of the network and impose constraints to the network output. Their experimental setup is about identical to ours except that they do not specify which part of Cityscapes is considered as the test set. Nonetheless, we include their results for comparison to put our work in a better perspective.

4.3.4.2 Comparison results

Table 4.2: Comparison results for adapting the FCN-8s model from SYNTHIA to Cityscapes.

Method %	IoU	SYNTHIA2Cityscapes Class-wise IoU															
		bike	fence	wall	t-sign	pole	mbike	t-light	sky	bus	rider	veg	bdlg	car	person	sidewalk	road
NoAdapt [86]	17.4	0.0	0.0	1.2	7.2	15.1	0.1	0.0	66.8	3.9	1.5	30.3	29.7	47.3	51.1	17.7	6.4
FCN Wld [86]	20.2	0.6	0.0	4.4	11.7	20.3	0.2	0.1	68.7	3.2	3.8	42.3	30.8	54.0	51.2	19.6	11.5
NoAdapt	22.0	18.0	0.5	0.8	5.3	21.5	0.5	8.0	75.6	4.5	9.0	72.4	59.6	23.6	35.1	11.2	5.6
NoAdapt (CC)	22.6	22.2	0.5	1.1	5.0	21.5	0.6	8.5	73.4	4.8	9.2	73.2	56.7	28.4	34.8	12.1	9.1
Ours (I)	25.5	16.7	0.8	2.3	6.4	21.7	1.0	9.9	59.6	12.1	7.9	70.2	67.5	32.0	29.3	18.1	51.9
Ours (CC+I)	27.3	31.2	1.3	3.9	6.0	19.4	2.1	9.2	61.2	11.2	7.4	68.3	65.1	41.4	29.3	18.9	60.6
SP Lndmk (CC)	23.1	0.0	0.0	0.0	0.0	0.0	0.0	0.0	82.6	27.8	0.0	73.1	67.9	40.7	5.8	10.3	62.2
SP (CC)	25.6	0.0	0.0	0.0	0.0	0.0	0.0	0.0	80.1	22.7	0.0	72.2	69.7	45.6	25.0	19.4	74.8
Ours (SP)	28.1	10.2	0.4	0.1	2.7	8.1	0.8	3.7	68.7	21.4	7.9	75.5	74.6	42.9	47.3	23.9	61.8
Ours (CC+SP)	28.9	17.7	0.5	0.5	3.4	10.9	1.8	5.4	73.4	17.6	9.9	76.8	74.5	43.7	44.4	22.4	59.6
Ours (I+SP)	29.0	13.1	0.5	0.1	3.0	10.7	0.7	3.7	70.6	20.7	8.2	76.1	74.9	43.2	47.1	26.1	65.2
Ours (CC+I+SP)	29.7	20.3	0.6	0.5	4.3	14.0	1.9	5.3	73.7	21.2	11.0	77.8	74.7	44.8	45.0	23.1	57.4

Table 4.3: Comparison results for adapting the FCN-8s model from GTA to Cityscapes.

Method %	IoU	GTA2Cityscapes Class-wise IoU																		
		bike	fence	wall	t-sign	pole	mbike	t-light	sky	bus	rider	veg	terrain	train	bdlg	car	person	truck	sidewalk	road
NoAdapt [86]	21.1	0.0	3.1	7.4	1.0	16.0	0.0	10.4	58.9	3.7	1.0	76.5	13	0.0	47.7	67.1	36	9.5	18.9	31.9
FCN Wld [86]	27.1	0.0	5.4	14.9	2.7	10.9	3.5	14.2	64.6	7.3	4.2	79.2	21.3	0.0	62.1	70.4	44.1	8.0	32.4	70.4
NoAdapt	22.3	13.8	8.7	7.3	16.8	21.0	4.3	14.9	64.4	5.0	17.5	45.9	2.4	6.9	64.1	55.3	41.6	8.4	6.8	18.1
NoAdapt (CC)	26.2	16.2	10.9	8.8	18.5	23.3	7.0	13.2	62.7	5.4	19.0	65.1	5.8	2.3	64.8	63.9	42.2	9.2	13.8	45.0
Ours (I)	23.1	9.5	9.4	10.2	14.0	20.2	3.8	13.6	63.8	3.4	10.6	56.9	2.8	10.9	69.7	60.5	31.8	10.9	10.8	26.4
Ours (CC+I)	28.5	7.2	9.4	11.1	13.4	23.1	9.6	15.1	64.6	5.9	15.5	71.1	10.3	3.9	67.7	62.3	43.0	14.0	23.0	71.6
SP Lndmk (CC)	21.6	0.0	0.0	0.0	0.0	0.0	0.0	0.0	82.4	9.1	0.0	74.4	22.2	0.0	70.3	53.1	15.3	11.2	6.9	65.8
SP (CC)	26.8	0.3	2.3	6.8	0.0	0.2	3.4	0.0	80.5	25.5	4.1	73.5	31.4	0.0	71.0	61.6	28.2	30.4	17.3	73.3
Ours (SP)	27.8	15.6	11.7	5.7	12.0	9.2	12.9	15.5	64.9	15.5	9.1	74.6	11.1	0.0	70.5	56.1	34.8	15.9	21.8	72.1
Ours (CC+SP)	30.2	10.4	13.6	10.3	14.0	13.9	18.8	16.5	73.6	14.1	9.5	79.2	12.9	0.0	74.3	63.5	33.1	18.9	27.5	70.5
Ours (I+SP)	28.9	14.6	11.9	6.0	11.1	8.4	16.8	16.3	66.5	18.9	9.3	75.7	13.3	0.0	71.7	55.2	38.0	18.8	22.0	74.9
Ours (CC+I+SP)	31.4	12.0	13.2	12.1	14.1	15.3	19.3	16.8	75.5	19.0	10.0	79.3	14.5	0.0	74.9	62.1	35.7	20.6	30.0	72.9

The overall comparison results of adapting from SYNTHIA to Cityscapes (SYNTHIA2Cityscapes) are shown in Table 4.2. We also present the results of adapting from GTA to Cityscapes (GTA2Cityscapes) in Table 4.3. Immediately, we note that all of our domain adaptation results are significantly better than those without adaptation (NoAdapt) in both tables.

We denote by (**Ours (I)**) the network regularized by the global label distributions over the target images. Although one may wonder that the image-wise label distributions are too abstract to supervise the pixel-wise discriminative network, the gain is actually significant. They are able to correct some obvious errors of the baseline network, such as the disproportional predictions about road and sidewalk (cf. the results of **Ours (I)** vs. NoAdapt in the last two columns of Tables 4.2 and 4.3).

It is interesting to see that both superpixel classification-based segmentation results (SP and SP Lndmk) are also better than the baseline network (NoAdapt). The label distributions obtained over the landmark superpixels boost the segmentation network (**Ours (SP)**) to the mean IoU of 28.1% and 27.8% when adapting from SYNTHIA and GTA, respectively, which are better than those by either superpixel classification or the baseline network individually. We have also tried to use the label distributions over all superpixels to train the network and yet observe little improvement. This is probably because it is too forceful to regularize the network output at every single superpixel especially when the estimated label distributions are not accurate enough.

The superpixel-based methods, including **Ours (SP)**, miss small objects, such as pole and traffic signs (t-sign) and instead are very accurate for categories like the sky, road, and building, which typically occupy larger image regions. In contrast, the label distributions on the images give rise to a network (**Ours (I)**) that performs better on the small objects than **Ours (SP)**. In other words, they mutually complement to some extent. Re-training the network by using the label distributions over both global images and local landmark superpixels (**Ours (I+SP)**), we achieve semantic seg-

mentation results on the target domain that are superior over using either to regularize the network.

Finally, we report the results of our method and its ablated versions (i.e., **Ours (I+SP)**, **Ours (I)**, and **Ours (SP)**) after we apply color constancy to the images (accordingly, the methods are denoted by **Ours (CC+I+SP)**, **Ours (CC+I)**, and **Ours (CC+SP)**). We observe improvements of various degrees over those before the color constancy. Especially, the best results are obtained after we apply the color constancy for adapting from both SYNTHIA and GTA.

4.3.4.2.1 Comparison with FCNs in the wild [86]

Although we use the same segmentation network (FCN-8s) as [86], our baseline results (NoAdapt) are better than those reported in [86]. This may be due to subtle differences in terms of implementation or experimental setup. For both SYNTHIA2Cityscapes and GTA2Cityscapes, we gain larger improvements (7.7% and 9%) over the baseline [86].

4.3.4.2.2 Comparison with learning domain-invariant features

In our first attempt to solve the domain adaptation problem for the semantic segmentation of urban scenes, we tried to learn domain invariant features following the deep domain adaptation method [115] for classification. In particular, we impose the maximum mean discrepancy [77] over the layer before the output. We name such network layer the feature layer. Since there are virtually three output layers in FCN-8s, we experiment with all three feature layers correspondingly. We have also tested the domain adaptation by reversing the gradients of a domain classifier [65]. However, none of these efforts lead to any noticeable gain over the baseline network, so the results are omitted.

4.3.4.3 *Confusion between classes*

While Tables 4.2 and 4.3 show the overall and per-class results, they do not tell the confusion between different classes. In this section, we provide the confusion matrices of some methods in order to provide more informative analyses about the results. Considering the page limit, we present in Figure 4.5 only the confusion matrices of NoAdapt and **Ours (CC+I+SP)** for SYNTHIA2Cityscapes and GTA2Cityscapes.

We find that a lot of objects are misclassified to the “building” category, especially the classes “pole”, “traffic sign”, “traffic light”, “fence” and “wall”. It is probably because those classes often show up beside buildings and they all have huge intra-class variability. Moreover, the “pole”, “traffic sign”, and others are very small objects compared to “building”. Some special care will be required to disentangle these classes from “building” in future work.

Another noticeable confusion is between the “train” and “bus” classes. After analyzing the data, we find that this is likely due to the lack of discrimination between the two classes by the datasets themselves, rather than the algorithms. In Figure 4.6, we visualize some trains and buses in the GTA dataset and some trains in the Cityscapes dataset. The difference between the trains and the buses turns out to be very subtle. We humans could make mistakes too if we do not pay attention to the rails (e.g., the train on the bottom right could easily be misclassified as a bus). This confusion between trains and buses could probably be alleviated if more training examples were supplied.

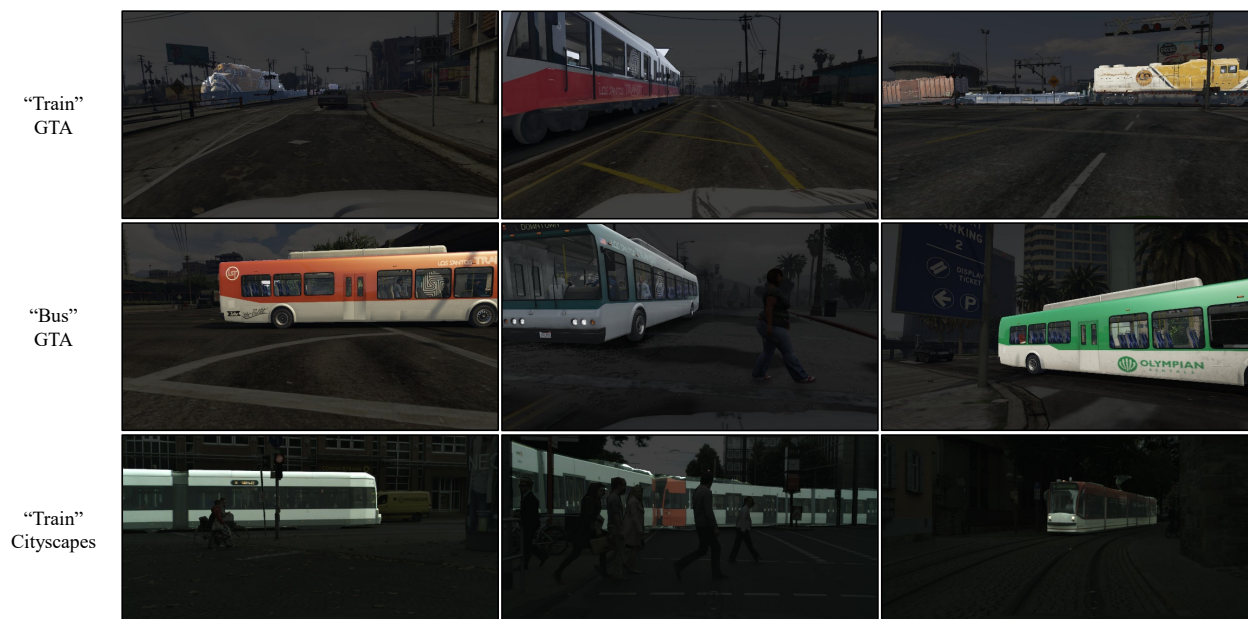


Figure 4.6: Some “train” and “bus” images from the Cityscapes and GTA datasets. We can see that the Cityscapes “trains” are visually more similar to the GTA “buses” than to the GTA “trains”.

Table 4.4: Results for the adaptation of FCN-8s from GTA to Cityscapes when we use handcrafted features instead of the CNN features.

Method	%	IoU	GTA2Cityscapes Class-wise IoU																		
			bike	fence	wall	t-sign	pole	mbike	t-light	sky	bus	rider	veg	terrain	train	bdg	car	person	truck	sidewalk	road
NoAdapt (CC)		26.2	16.2	10.9	8.8	18.5	23.3	7.0	13.2	62.7	5.4	19.0	65.1	5.8	2.3	64.8	63.9	42.2	9.2	13.8	45.0
Ours (CC+BOW)		27.9	13.8	14.0	9.6	17.9	23.9	6.4	16.7	64.6	3.0	18.0	69.1	7.0	2.4	69.2	60.1	44.0	10.7	19.1	60.8
Ours (CC+FV)		28.1	13.3	10.5	12.8	18.6	24.4	5.1	10.8	63.5	1.7	14.6	73.5	10.0	0.3	71.6	66.6	40.8	6.1	11.5	79.0
Ours (CC+BOW+FV)		28.3	15.3	13.3	11.6	18.5	25.1	6.7	16.8	66.5	2.9	18.4	72.2	8.7	2.6	70.0	59.4	43.9	10.8	19.1	56.8

4.3.5 Representations of the superpixels

One may wonder how the representations of the superpixels affect the overall domain adaptation results. We conduct detailed analyses in this section to answer this question. Our main results

(Tables 4.2 and 4.3) are obtained by representing the landmark superpixels with the networks pre-trained on PASCAL CONTEXT [127]. We are interested in examining whether or not such high-level semantic representations of the superpixels are necessary. Hence, we compare the high-level superpixel descriptors with the following low-level features.

4.3.5.1 Handcrafted features

BOW. We encode the superpixels with the bag-of-words (BOW)-SIFT features. We first extract dense SIFT features [116] from the input image and then encode those of each superpixel into a 100D BOW vector. The dictionary for the encoding is obtained by K-means clustering.

FV. We also replace the image-level CNN features with the Fisher vectors (FV) [150, 149] for estimating the label distribution of a target image. FV encodes the SIFT features per image into a fixed-dimensional descriptor through a Gaussian mixture model, which has 8 components in this work. An image is then represented by a 2048D vector. We train the dictionary for BOW and the Gaussian mixture model for FV using the SIFT features of the GTA dataset only.

Table 4.4 shows the GTA2Cityscapes results using the handcrafted features. We denote the resulting methods respectively by **Ours (CC+BOW)**, **Ours (CC+FV)**, and **Ours (CC+BOW+FV)**. It is interesting to see all of these outperform the baseline **NoAdapt (CC)**, indicating that our curriculum domain adaptation method is able to leverage the handcrafted features as well.

4.3.5.2 VGG features

We also test the VGG features [175] of the `block5_conv4` and `block4_conv4` layers. The network is pre-trained on ImageNet and does not bring in any extra knowledge as the segmentation

networks are also pre-trained with ImageNet. In order to extract the features for each superpixel, we average-pool the activations per channel within the superpixel.

With the new VGG features, the superpixel classification accuracy on the Cityscapes validation set is 76%, a 5% boost from the accuracy of the PASCAL-CONTEXT features. Moreover, the top 30%, which are the landmark superpixels used in our experiments, are labeled with up to 93% accuracy (vs. 88% with the PASCAL-CONTEXT features). Thanks to the boost in the classification accuracy of the landmark superpixels, we also observe a 0.6% gain in mIoU (from 29% to 29.6%) on the domain adaptation from SYNTHIA to Cityscapes. These results imply that the representations of the superpixels do influence the final results, but the representations via an extra knowledge base are not necessarily advantageous; the final results with the ImageNet-pretrained VGG features are superior over those with the PASCAL-CONTEXT features.

4.3.6 Granularity of the superpixels

In this section, we study what is a proper granularity of the superpixels. Intuitively, small superpixels are fine-grained and precisely track object boundaries. However, they are less discriminative as a result. What is a proper granularity of the superpixels? How sensitive to granularity are the results? To quantitatively answer these questions, we vary the number of superpixels per image to examine their effects on the semantic segmentation results. The adaptation from GTA of the **SP (CC)** method, described in Section 4.3.4.1, is reported in Table 4.5 with various numbers of superpixels per image. In general, the performance increases as the number of superpixels grows until it reaches 400 per image.

We present the classification accuracy of the top $x\%$ superpixels in Figure 4.3.6, where $x = 0, 20, \dots, 100$. We can see that the accuracies are always more than 90% when we keep the top 5% to 50% of superpixels per image — in our experiments, we keep the top 30%. The accuracies

in keeping all superpixels (top 100%) are significantly lower, indicating that it is not a good idea to use all superpixels to guide the training of the neural networks.

Table 4.5: Results for the adaptation of FCN-8s from GTA to Cityscapes when we use different numbers of superpixels per image. Here the images are pre-processed with color constancy.

SP # per image	IoU	GTA2Cityscapes Class-wise IoU																		
		bike	fence	wall	t-sign	pole	mbike	t-light	sky	bus	rider	veg	terrain	train	bdlg	car	person	truck	sidewalk	road
50	22.9	0.0	0.0	1.5	0.0	0.0	0.0	0.0	80.1	21.9	0.0	69.3	26.2	0.0	66.3	54.9	9.9	18.3	12.6	74.6
100	26.8	0.3	2.3	6.8	0.0	0.2	3.4	0.0	80.5	25.5	4.1	73.5	31.4	0.0	71.0	61.6	28.2	30.4	17.3	73.3
200	27.3	0.2	2.2	7.2	0.0	0.8	3.0	0.0	80.5	24.4	3.8	75.9	32.9	0.0	72.5	63.7	31.1	27.9	18.4	74.2
400	27.4	0.4	3.8	7.2	0.0	1.1	1.7	0.0	80.7	23.2	3.7	76.9	33.3	0.0	72.5	63.8	33.1	26.9	18.4	73.3

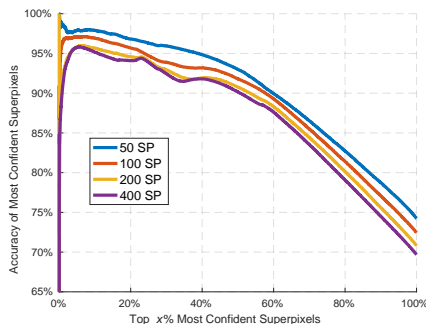


Figure 4.7: We evaluate how many superpixels are accurate in the top $x\%$ confidently predicted superpixels. The experiments are conducted on the validation set of Cityscapes with color constancy.

4.3.7 Domain adaptation experiments using ADEMXP

Our approach is agnostic to the base semantic segmentation neural networks. In this section, we further investigate a more recent network, ADEMXP [199], which is among the few top performing methods on the Cityscapes challenge board. Our experimental setup in this section resembles that of Section 4.3.4 except that we replace FCN-8s with the ADEMXP net. In particular,

we reimplement the A1 model of ADEMXPAPP using the Theano-Keras framework. However, we remove the batch normalization layers in our implementation due to their extensive GPU memory consumption. We follow the authors’ suggestions otherwise and initialize the network with the weights pre-trained on Imagenet. We set the size of the mini-batch to six, three images from the source domain and the other three from the target domain.

Table 4.6 shows the comparison results for the ADEMXPAPP net. We see that it indeed achieves much better results than FCN-8s in general. Nonetheless, the relative trend of our approach against the others remains the same for this ADEMXPAPP net.

Table 4.6: Results for the adaptation of ADEMXPAPP [199] from GTA to Cityscapes. The ADEMXPAPP net is a more powerful semantic segmentation network than FCN-8s.

Method %	IoU	GTA2Cityscapes Class-wise IoU																		
		bike	fence	wall	t-sign	pole	nbike	t-light	sky	bus	rider	veg	terrain	train	bdlg	car	person	truck	sidewalk	road
FCN (CC)	26.2	16.2	10.9	8.8	18.5	23.3	7.0	13.2	62.7	5.4	19.0	65.1	5.8	2.3	64.8	63.9	42.2	9.2	13.8	45.0
ADEMXPAPP (CC)	30.0	8.3	10.6	15.5	16.5	23.3	11.7	21.9	66.5	10.7	12.6	74.2	14.1	3.3	70.2	58.4	43.1	14.3	24.2	70.2
FCN (CC+SP)	30.2	10.4	13.6	10.3	14.0	13.9	18.8	16.5	73.6	14.1	9.5	79.2	12.9	0.0	74.3	63.5	33.1	18.9	27.5	70.5
ADEMXPAPP (CC+SP)	34.0	8.8	12.4	18.4	15.3	22.5	16.6	18.5	73.7	24.5	10.9	76.7	22.9	0.1	74.3	72.3	40.3	21.3	32.9	83.1
FCN (CC+I)	28.5	7.2	9.4	11.1	13.4	23.1	9.6	15.1	64.6	5.9	15.5	71.1	10.3	3.9	67.7	62.3	43.0	14.0	23.0	71.6
ADEMXPAPP (CC+I)	31.4	24.2	13.3	19.0	11.2	26.2	10.0	8.2	62.4	9.0	18.8	74.0	14.5	8.1	68.4	70.2	41.4	16.3	26.9	73.6
FCN (CC+I+SP)	31.4	12.0	13.2	12.1	14.1	15.3	19.3	16.8	75.5	19.0	10.0	79.3	14.5	0.0	74.9	62.1	35.7	20.6	30.0	72.9
ADEMXPAPP (CC+I+SP)	35.7	9.6	13.6	21.9	14.2	25.0	15.7	19.2	72.3	22.1	18.1	77.7	19.7	14.5	77.8	73.3	46.4	17.4	33.8	85.0

4.3.8 What is the “market value” of the synthetic data?

Despite the positive results thus far for our curriculum domain adaptation from simulation to reality for the semantic segmentation of urban scenes, we argue that the significance of our work is in its capability to complement the training sets of real data, rather than replacing them. In the long run, we expect that learning from both simulation and reality will alleviate the strong dependency of deep learning models on massive labeled real training data. Therefore, it is interesting to evaluate

the “market value” of the synthetic data in terms of the labeling effort: how many real training images can the GTA or SYNTHIA dataset obviate in order to achieve about the same level of segmentation accuracy?

In order to answer the above question, we design the following experiment. We train two versions of the VGG-19-FCN-8s network. One is trained on a portion of annotated Cityscapes images while another one is trained using the same portion of Cityscapes images plus the entire SYNTHIA dataset. The subset is sampled from 2380 Cityscapes training images in the experiment since the remaining ones are reserved for validation. We report both models’ performances under different percentages subsampled from Cityscapes.

Table 4.7 presents the results. First of all, it is somewhat surprising to see that even as few as five Cityscapes training images added to the SYNTHIA training set can significantly boost the results obtained from only synthetic images (from 22.0% to 33.8%). Second, the “N/A” results in the table mean that the corresponding neural networks either give rise to random predictions or have numerical issues. Note that such phenomena happen until there are more than 450 target images for the training without any synthetic images, implying that the “market value” of the SYNTHIA training set is at least worth 450 well-labeled real images. Actually, if we compare the results of the two rows, the network trained from the mixed training set outperforms the one from the real images up to the 50% mix. In other words, augmenting the Cityscapes training set with the SYNTHIA training set improves performance when the Cityscapes training set is smaller than 1000 images.

Table 4.7: IoUs after mixing different percentages of Cityscapes images into the SYNTHIA training dataset (SYN+CS), and of models trained with different percentages of Cityscapes images without any SYNTHIA images (CS only).

# CS images	None	5 images	1%	2.5%	5%	10%	20%	50%	100%
SYN+CS	22.0%	33.8%	38.4%	41.0%	43.0%	46.5%	48.5%	53.2%	57.3%
CS only	N/A	N/A	N/A	N/A	N/A	N/A	38.4%	52.2%	57.8%

4.4 Review of the recent works on domain adaptation for semantic segmentation

After our work [210] published in the IEEE International Conference on Computer Vision in 2017, there has been notably a rich line of works tackling the same problem, i.e., domain adaptation for semantic segmentation of urban scenes by adapting from synthetic imagery to real images. Some of them have reported very good results. Since our approach is “orthogonal” in some sense to these others, one may achieve even better results by fusing our method with these new ones. In this section, we give a comprehensive review of the new methods and also present the results of fusing ours with some of them. Most of the methods resort to adversarial training to reduce the domain discrepancy. We review such methods first, followed by the others.

4.4.1 Adversarial training based methods

If an adversarial classifier fails to differentiate the data instances of the source domain and the target domain, the discrepancy between the two should have been eliminated in a certain sense. Many methods depend on this principle and differ on how they incorporate it into the training of the segmentation networks.

FCNs in the wild [86]. Hoffman et al. generalize FCNs [114] from the source domain of synthetic imagery to the target domain of real images for the semantic segmentation task. They employ a pixel-level adversarial loss to enforce the network to extract domain-invariant features.

CyCADA [85]. The main idea is to transform the synthetic images of the source domain to the style of the target domain (real images) using CycleGAN [213] before feeding the source images to the segmentation network. CycleGAN and the segmentation network are trained simultaneously.

ROAD [37]. In the Reality Oriented Adaptation (ROAD) method [37], two losses are proposed to align the source and the target domain. The first one is called target guided distillation, which is a loss for regression from the segmentation network’s hidden layer activation of the source domain to the image features of the target domain. Here, the image features are obtained by a classifier pre-trained on ImageNet. The other loss takes care of the spatial-aware adaptation. The feature map of either a source image or a target image is partitioned into non-overlapping grids. After that, a maximum mean discrepancy loss [86] is introduced over each grid.

MCD [165]. The Maximum Classifier Discrepancy (MCD) resembles the recently popularized generative adversarial methods [65]. It learns two classifiers from the source domain and maximizes their disagreement on the target images in order to detect target examples that fall out of the support of the source domain. After that, it updates the generator to minimize the two classifiers’ disagreement on the target domain. By alternating the two steps in the training, it ensures that the generator gives rise to feature representations over which the source and the target domains are well aligned.

LSD [167]. This work is an adversarial domain adaptation network built upon an auto-encoder network. The network takes as input both source and target images and reconstructs them due to an auto-encoder loss. Meanwhile, an intermediate layer is connected to the segmentation network whose loss is defined using the labeled source images.

AdaptSegNet [188]. Similar to FCN in the wild [86], this work also employs the adversarial feature learning over the base segmentation model. Instead of having only one discriminator over the feature layer, Tsai et al. propose to install another discriminator on one of the intermediate layers as well. Essentially, features of different scales are forced to align.

CGAN [88]. This work proposes to add a fully convolutional auxiliary pathway to inject random

noise into the source domain. Hence, what the segmentation network receives is the source images with perturbations. The authors found such a structure, which is motivated by the conditional GAN, greatly boosts the adaptation performance.

ADR [164]. In ADR, the pixel classifier also serves as the domain classifier. It employs dropout to avoid generating features near the classification boundaries so as to avoid ambiguity to the classifier.

NMD [38]. Besides the global adversarial feature learning module, NMD proposes a local class-wise adversarial loss over image grids. Each image grid is associated with a label distribution. The class-wise adversarial learning then tries to differentiate the source domain's label distributions from those of the target domain due to the semantic segmentation network.

DAM [90]. Huang et al. train two separate networks for the source and target domains, respectively. Since there is no segmentation annotation in the target domain, the target-domain network is trained by both regressing to the source network's weights and an adversarial loss over every layer of the two networks.

FCAN [211]. Zhang et al. apply the adversarial loss to the lower layers of the segmentation network in addition to the common practice of using it over the last one or a few layers. The intuition is that it plays a complementary role because the lower layers mainly capture the appearance information of the images.

DCAN [198]. DCAN is a two-stage end-to-end network. In the first stage, it is adversarially trained to transfer the source (synthetic) images to the target (real) style. In its second stage, adversarially learning aligns the intermediate features of the two domains. Unlike the other methods, its adversarial alignment only accounts for channel-wise features.

I2I [129]. Similar to DCAN, I2I is another domain adaptation method that learns domain agnostic features by training both an image translation network and a segmentation network.

CLoss [215]. Zhu et al. introduce a conservative loss in addition to the adversarial training. The conservative loss prevents overfitting the model to the source domain by penalizing overly confident source domain predictions during training.

4.4.2 Other methods

Adversarial training is so popular that there are only a few methods that are not of the adversarial vein for the domain adaptation of semantic segmentation.

IBN [139]. Pan et al. show that a careful balance between the instance normalization and batch normalization can enhance a neural network’s cross-domain generalization.

EUSD [166]. Arguing that object detectors have better generalization capacity in detecting foreground objects (e.g., car, pedestrian, etc.) than the background, Saleh et al. propose a simple yet powerful domain generalization segmentation framework by fusing Mask-RCNN’s detection results of the foreground [81] and DeepLab’s segmentation results of the background [33].

DAN [157]. As normalization (e.g., batch normalization) plays a key role in many neural semantic segmentation networks, DAN improves their performance in the target domain by simply replacing the normalization parameters with the statistics of the target domain. This change of normalization boosts the network’s results in the target domain.

CBST [217]. Similar to our approach, this paper proposes a curriculum learning method for the domain adaptation of semantic segmentation. They introduce a class-balanced self-training strategy: the most confident predictions by a model on the unlabeled instances are likely to be correct. In each round of the training process, CBST selects some of the most confident predictions on the target images and include them in the training set of the next round.

4.4.3 Results reported in other papers

We summarize in Table 4.8 the results reported in these papers, along with ours. Immediately, we see that the backbone network has a huge impact on both the source-only performance without applying any adaptation techniques and the relative gain after adaptation. For instance, the performance gain of MCD jumps from 3.9% to 17.5% after switching the backbone network from VGG to DRN [206].

Another interesting observation is that the results of no adaptation vary greatly even when the same backbone network (e.g., VGG16) is used, implying that subtle changes to the implementation (e.g., removing batch normalization in order to save computation cost) can result in big differences among the final results. Through private communications with some authors of these papers, we also learned that the image resolution is another key factor. In general, higher resolution of the input image gives rise to better results no matter with or without domain adaptation.

4.4.4 The existing methods and ours are complementary

In this section, we provide experimental evidence to show that our method is complementary to most existing methods. A late fusion of our method with another can yield better results than either individually. This is not surprising as our curriculum domain adaptation approach guides the network towards the target domain by inferring properties of the labels, whereas most existing methods learn domain-invariant features or image styles.

First of all, we analyze the class-wise prediction accuracy (evaluated by mIoU) of different methods on the target domain. Figure 4.8 shows the pairwise comparison between our method and some other representative methods. The entry (i, j) shows the number of classes by which the i -th method outperforms the j -th on the target domain. This table is derived from the results

of GTA2Cityscapes. Our method here is the GTA model trained without color constancy. The class-wise comparison matrix is best understood if we recall the results in Table 4.8. Take I2I for instance. While it outperforms ours by 2.9% in mIoU, ours is superior over I2I in 10 out of the 19 classes. We can draw similar observations for the other methods. To this end, we find that our approach is genuinely complementary to the other methods. Finally, we find that CBST is complementary to other adversarial training based methods as well since it is in the same vein as our curriculum domain adaptation strategy.

It is worth emphasizing that the class-wise complementary relationship between these methods is only one of the possible perspectives for the analyses. More fine-grained analyses, for example image-wise, may reveal further insights about the methods.

Equipped with the class-wise comparison between any pair of methods, we design a simple late fusion scheme to ensemble two models. We first identify the classes for which one model gives rise to more accurate prediction than the other model by using the validation set. Given a test image, we keep its pixel-wise labels of those classes predicted by the former model. For the remaining pixels, we label them by the latter model. We apply this late fusion scheme to CYCADA and our approach. For GTA2Cityscapes, the mIoUs of CYCADA and ours (without color constancy) are 32.5% and 28%, respectively. In contrast, the late fusion of the two leads to an mIoU of 34.3% which is higher than either of them. This result and the analysis shown in Figure 4.8 clearly evidence that most of the existing methods are complementary to ours for the domain adaptation of the semantic segmentation task.

Table 4.8: Comparison with the recent works published after the conference version of our approach [210].

Method	Backbone Network	Base X Adapt ✓	SYN2CS		GTA2CS		Main idea	Code
			IoU	Gain	IoU	Gain		
CyCADA [85]	VGG-16	×	-	-	17.9	-	Adversary	Link
		✓	-	-	35.4	17.5		
	DRN-26	×	-	-	21.7	17.8		
		✓	-	-	39.5	-		
ROAD [37]	VGG-16	×	25.4	10.8	21.9	13.0	Adversary	-
		✓	36.2	-	35.9	-		
MCD [165]	VGG-16	×	-	-	24.9	3.9	Adversary	Link
		✓	-	-	28.8	-		
	DRN-105	×	23.4	13.9	22.2	17.5		
		✓	37.3	-	39.7	-		
LSD [167]	VGG-16	×	26.8	9.3	29.6	7.5	Adversary	Link
		✓	36.1	-	37.1	-		
AdaptSegNet [188]	VGG-16	×	-	-	-	-	Adversary	Link
		✓	37.6**	-	35.0	-		
	ResNet-101	×	38.6**	-	36.6	5.8		
		✓	46.7**	9.1	42.4	-		
FCAN [211]	ResNet-101	×	-	-	29.2	17.4	Adversary	-
		✓	-	-	46.6	-		
ADR [164]	ResNet-50	×	-	-	25.3	8.0	Adversary	-
		✓	-	-	33.3	-		
I2I [129]	ResNet-34	×	-	-	21.1	10.7	Adversary	-
		✓	-	-	31.8	-		
	DenseNet-121	×	-	-	29.0	6.7		
		✓	-	-	35.7	-		
DCAN [198]	VGG-16	×	25.9	9.5	27.8	8.4	Adversary	-
		✓	35.4	-	36.2	-		
	ResNet-101	×	28.0	8.5	29.8	8.7		
		✓	36.5	-	38.5	-		
	PSPNet	×	29.5	8.9	33.3	8.4		
		✓	38.4	-	41.7	-		
DAM [90]	VGG-16	×	22.0 [†]	8.7 [†]	18.8	13.8	Adversary	Link
		✓	30.7	-	32.6	-		
	DRN-26	×	-	-	-	-		
		✓	-	-	40.2	-		
	ERFNet	×	-	-	15.8	15.5		
		✓	-	-	31.3	-		
CGAN [88]	VGG-16	×	17.4	23.8	21.1	23.4	Adversary	-
		✓	41.2	-	44.5	-		
NMD [38]	Dilation Frontend	×	30.7**	-	-	-	Adversary	-
		✓	35.7**	5.0	-	-		
C-Loss [215]	VGG-16	×	24.9	9.3	30.0	8.1	Adversary	-
		✓	34.2	-	38.1	-		
FCN Wild [86]	Dilation Frontend	×	17.4	2.8	21.1	6.0	Adversary	-
		✓	20.2	-	27.1	-		
CBST [217]	VGG-16	×	22.6	12.8	24.3	11.8	Self-Training	Link
		✓	35.4	-	36.1	-		
	ResNet-38	×	29.2	13.3	35.4	11.6		
		✓	42.5	-	47.0	-		
IBN [139]	ResNet-50	×	-	-	22.2	7.4	Normalization	Link
		✓	-	-	29.6	-		
EUSD [166]	DeepLab	×	-	-	31.3	11.2	Ensemble Detector & Segmenter	-
	Mask R-CNN	✓	-	-	42.5	-		
DAN [157]	ResNet-50	×	-	-	34.8	3.4	Normalization	Link
		✓	-	-	38.2	-		
Ours	VGG-19	×	22.0	7.7	22.3	9.1	Curriculum	Link
		✓	29.7	-	31.4	-		
	VGG-19***	×	22.0	7.6	-	-		
		✓	29.6	-	-	-		
	DRN-26	×	21.9	6.3	-	-		
		✓	28.2	-	-	-		
	ADEMXAPP	×	-	-	30.0	5.7		
		✓	-	-	35.7	-		

* DAM did not report the baseline performance of the SYNTHIA2Cityscapes experiment. Since their model is fine-tuned from our baseline, we report our baseline performance instead.

** NMD and AdaptSegNet used 13 SYNTHIA classes in their experiments instead of the commonly used 16.

*** We do not use any external information.

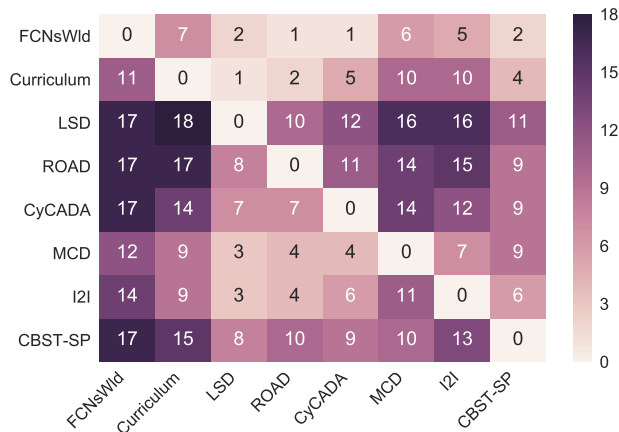


Figure 4.8: Pairwise comparison between different domain adaptation methods for the semantic segmentation task. The entry (i, j) of this table is the number of classes by which the i -th method outperforms the j -th. The results are obtained on GTA2Cityscapes. Our method is labeled *Curriculum*.

4.5 Summary

In this chapter, we propose a curriculum domain adaptation approach for the semantic segmentation of urban scenes. We learn to estimate the global label distributions over the target images and local label distributions over the superpixels of the target images. These tasks are easier to solve than the pixel-wise label assignment. We then use their results to effectively regularize the training of the semantic segmentation networks such that their pixel-wise predictions are consistent with the global and local label distributions. We experimentally verify the effectiveness of our approach by adapting from the source domain of synthetic images to the target domain of real images. Our method outperforms several competing baselines. Moreover, we report several key ablation studies that allow us to gain more insights about the proposed method. We also check the class-wise confusion matrices and find that some of the classes (e.g., train and bus) are almost indistinguishable in the current datasets, indicating that better simulation or more labeled real ex-

amples are required in order to achieve better segmentation results. In future work, we will explore more target properties that possess the same form as the global and local label distributions — they are easier to solve than the pixel-wise label prediction and meanwhile can be written as a function of the pixel-wise labels. We also would like to look into the possibility of directly applying our domain adaptation framework to virtual autonomous driving environments such as DeepGTAV [161] and AirSim [170].

CHAPTER 5: LEARNING TRANSFERABLE ADVERSARIAL CAMOUFLAGES

5.1 Problem Introduction

Is it possible to paint a unique pattern on a vehicle’s body and hence hide it from being detected by surveillance cameras? We conjecture the answer is affirmative mainly for two reasons. First, deep neural networks will be widely used in modern surveillance and autonomous driving systems for automatic vehicle detection. Second, unfortunately, these neural networks are intriguingly vulnerable to adversarial examples [8].

[181] found that adding imperceptible perturbations to clean images can result in the failure of neural networks trained for image classification. This motivates a rich line of work on developing defense techniques for the neural networks [8] and powerful attack methods to defeat those defenses [12]. Moreover, the adversarial attack has been extended to other tasks, such as semantic segmentation [11], object detection [201], image captioning [31], etc.

This chapter contains previously published materials from “CAMOU: Learning Physical Vehicle Camouflages to Adversarially Attack Detectors in the Wild” by Yang Zhang, Hassan Foroosh, Philip David and Boqing Gong, published in Proceedings of the International Conference on Learning Representations in 2019.

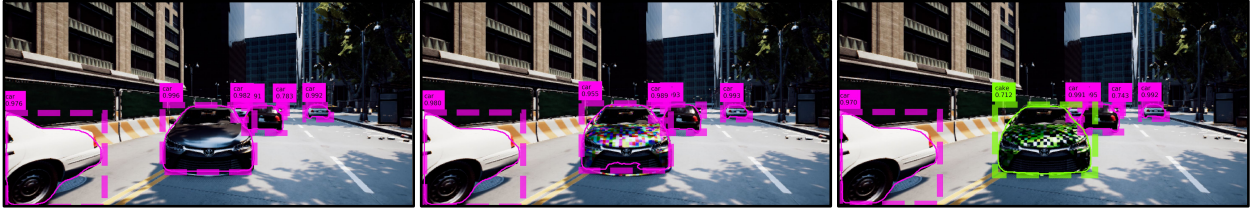


Figure 5.1: A Toyota Camry XLE in the center of the image fools the Mask R-CNN object detector after we apply the learned camouflage to it (on the right), whereas neither plain colors (on the left) nor a random camouflage (in the middle) is able to escape the Camry from being detected.

It is worth noting that the adversarial examples in the works mentioned above are not physical, i.e., the adversary directly manipulates image pixels. Although it is arguably more challenging to create physical adversarial objects than to produce adversarial images, some existing works have shown promising results with adversarial patches [25], stop signs [51, 36], and small objects like baseballs and 3D turtle models [13].

To this end, we are reasonably optimistic about designing a special pattern to camouflage a 3D car, in order to make it difficult to detect by the deep learning based vehicle detectors.

It is undoubtedly challenging to run experiments in the real world considering financial and time constraints. In this chapter, we instead demonstrate results using a simulation engine [2] with a high-fidelity 3D sedan model and a 3D SUV. Fig. 5.1 shows that the vehicle in the simulation is photo-realistic such that, even covered with random camouflage, it can still be detected by the Mask R-CNN detector [81] trained on COCO [112].

The simulation engine enables us to test the physically adversarial cars under a considerable spectrum of environmental conditions: lighting, backgrounds, camera-to-object distances, viewing angles, occlusions, etc. In contrast, existing experiments on physical adversarial attacks are all executed in simplified scenarios. [51], [50], and [36] attack neural classifiers and detectors of stop

signs. While projective transformations could be used to render the planar stop signs to various images, it is more involved to image the nonplanar 3D vehicles considered in this chapter; we learn a neural approximation function instead. [13] synthesizes objects (e.g., baseball, turtle, etc.) which are adversarial within a small range of camera-to-object distances and viewing angles.

Given a 3D vehicle model in the simulation engine, we learn a camouflage for it by following the expectation over transformation (EoT) principle first formalized by [13]. The main idea is to consider a variety of transformations under which the camouflage can consistently hide the vehicle from a neural detector. A transformation imitates the imaging procedure and produces an image of the 3D vehicle model in the simulated environment. If a camouflage works under many transformations seen in the training phase, it is expected to also generalize to unseen transformations in the test phase.

One of the major challenges is that the simulator’s image generation procedure is non-differentiable. A seemingly plausible solution is to train a neural network to approximate this procedure. The network takes as input the environment, a camouflage pattern, and the 3D vehicle model and outputs an image as close as possible to the one rendered by the simulator. Although this approach is viable, it is extremely difficult to generate high-resolution images. State-of-the-art methods (e.g., RenderNet [132]) can only generate simple 3D objects without any backgrounds.

We tackle the above challenge by drawing the following observation. In EoT [13, 36], the gradients propagate back to the physical object from the detector/classifier’s decision values. If we jointly consider the object detector and the imaging procedure of the simulator as a whole black box, it is easier to learn a function to approximate this black box’s behavior than to train the image generation neural network. Hence, we learn a substitute neural network which takes as input a camouflage, the vehicle model, and the environment and outputs the vehicle detector’s decision value. Equipped with this substitute network, we can readily run the EoT algorithm [13] over our

simulator in order to infer an adversarial camouflage for the vehicles.

Finally, we make some remarks about the significance and potential impact of this work. In the real world, multiclass visual object detection neural networks [81, 153] have become the cornerstone of multiple industrial applications, such as surveillance systems [130], autonomous driving [91], and military systems [145]. Among these applications, cars are one of the most crucial objects. Attacking vehicle detectors in the physical world will be enormously valuable and impactful from the perspective of the malicious adversaries. Compared with the stop sign, it is legal in the United States to paint a car while defacing a stop sign is criminal. This poses a more significant threat to autonomous driving systems since anyone has access to perturb public machine learning based systems legally. This observation motivates us to focus our approach on cars. We limit our camouflage within the legally paintable car body parts, which means that we will leave discriminative visual cues, such as the tires, windows, grille, lights, etc. unaltered for the detectors.

5.2 Learning Objective

In this chapter, we investigate *physical adversarial attack* on state-of-the-art neural network based object detectors. The objective is to find a camouflage pattern such that, when it is painted on the body of a vehicle, the Mask R-CNN [81] and YOLO [153] detectors fail to detect the vehicle under a wide spectrum of variations (e.g., locations, lighting conditions, viewing angles, etc.). We learn the camouflage in a black-box fashion, without the need for accessing the detectors' network architecture or weights.

Expectation-over-transformation (EoT). We formalize the physical adversarial attack problem with the EoT framework [13]. Denote by t a transformation which converts a camouflage pattern c to a real photo. Such a transformation actually represents an involved procedure: paint the

pattern to a vehicle’s body, drive the vehicle to a location, configure a camera, and take a picture of the vehicle. The transformation conveniently abstracts away enormous factors (e.g., quality of painting, camera, etc.), each of which could play a role in this procedure. Denote by $V_t(c)$ the detection score (e.g., by Mask-RCNN) over an image which is a result of a transformation t and a camouflage c . The physical adversarial attack problem is then described by

$$\arg \min_c \mathbb{E}_{t \sim \mathcal{T}} V_t(c) \quad (5.1)$$

where \mathcal{T} is a distribution over all the possible transformations $\{t\}$. In other words, we search for a camouflage c that minimizes the vehicle detection score in expectation.



Figure 5.2: Example procedure of a transformation t . From left to right: a 16×16 camouflage, a high-fidelity vehicle, a location in our simulated environment, and an image due to this transformation over the camouflage.

Transformation in simulation. Due to financial and time constraints, we conduct our study with the photo-realistic [2] 4 game engine. It supplies us sufficient configuration parameters, such as the resolution and pattern of the camouflage, 3D models of vehicles, parameters of cameras and environments, etc. Fig. 5.2 shows a camouflage pattern, a high-fidelity 3D model of Toyota Camry, a corner of the virtual city used in this chapter, and finally the picture taken by a camera after we apply the camouflage to the Camry and drive it to that corner — in other words, the rightmost image is the result of a certain transformation t acting on the three images on the left of Fig. 5.2.

5.3 Approach

In this section, we present two key techniques for solving the problem $\min_c \mathbb{E}_t V_t(c)$ (cf. Eq. (5.1) and the text there). One is to estimate the expectation \mathbb{E}_t by an empirical mean over many transformations. The other is to train a neural network to clone the joint behavior $V_t(c)$ of the black-box detector and the non-differentiable simulator.

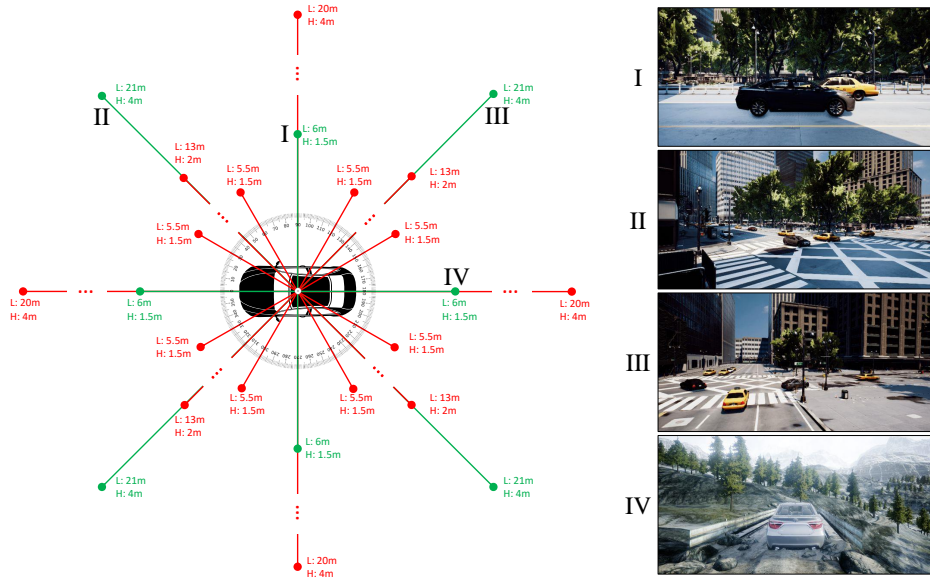


Figure 5.3: The camera setup as well as some exemplar locations. The cameras depicted in the **green** color are used to learn the camouflage while those in **red** are unseen cameras used for testing the camouflage’s generalization. (H: camera height, L: vehicle-to-camera distance.)

5.3.1 Sampling transformations to estimate \mathbb{E}_t

Recall that a transformation t specifies a particular procedure from applying the camouflage c to a vehicle until the corresponding image captured by a camera. In the context of the simulation engine, the camouflage c is first programmed as textures, which are then warped onto the 3D model of an object. The simulator can teleport the object to different locations in the environment. The

simulator also has multiple cameras to photograph the teleported object from various distances and viewing angles.

We identify some key factors involved in this procedure, including vehicle, location, camera-to-object distance, and viewing angle. The combinations of them also give rise to variations along other dimensions. For instance, the lighting condition changes from one location to another. The vehicle of interest is occluded to different degrees in the images captured by the cameras. Denote by T_S all the sampled transformations that are used for learning the camouflage.

Fig. 5.3 illustrates the positions where the cameras are placed. We can see how the viewing angles and camera-to-object distances vary. The ones shown in the green color are used in the training. We left out some cameras shown in the red color for the testing. Since it is computationally expensive to utilize the cameras, we randomly arrange them in different heights and distances to create as much variations as possible instead of traversing all possible height-distance combinations.

5.3.2 Learning a clone network $V_\theta(c, t)$ to approximate $V_t(c)$

If we unroll the detection score $V_t(c)$, it contains two major components. The first renders an image based on the camouflage c by following the procedure specified by the transformation t . The second obtains the detection score of a vehicle detector (e.g., Mask-RCNN) over the rendered image. The first component is non-differentiable while the second one could be a black box in practice. Therefore, we propose to consider them jointly as a single black box. Furthermore, we learn a clone neural network $V_\theta(c, t)$ to imitate the input-output behavior of this extended black box. As the transformation t itself is involved and hard to represent, we instead input its consequence to the network: a background image and the cropped foreground of the vehicle.

Fig. 5.4b shows the network architecture. It takes as input a camouflage pattern c , the background

image due to a sampled transformation t , and the cropped foreground. The network $V_\theta(c, t)$ has a single output to approximate the detection score $V_t(c)$.

To this end, we are ready to write down an approximate form of problem (5.1),

$$\arg \min_c \frac{1}{|T_S|} \sum_{t \in T_S} V_\theta(c, t). \quad (5.2)$$

Thanks to the differentiable network $V_\theta(c, t)$ with respect to the camouflage c , we can solve the problem above by standard (stochastic) gradient descent.

It is important to note that the fidelity of problem (5.2) depends on the size and diversity of the sampled set of transformations T_S as well as the quality of the clone network $V_\theta(c, t)$. It is straightforward to generate a large training set for the clone network by randomizing the camouflages and transformations and “labeling” the resulting images with the detection scores. However, if the optimal camouflage is unfortunately not covered by this training set, a discrepancy would occur when we solve problem (5.2). In other words, the network may fail to well approximate the detection scores at the region around the optimal camouflage. We address this discrepancy by an alternative learning algorithm as follows.

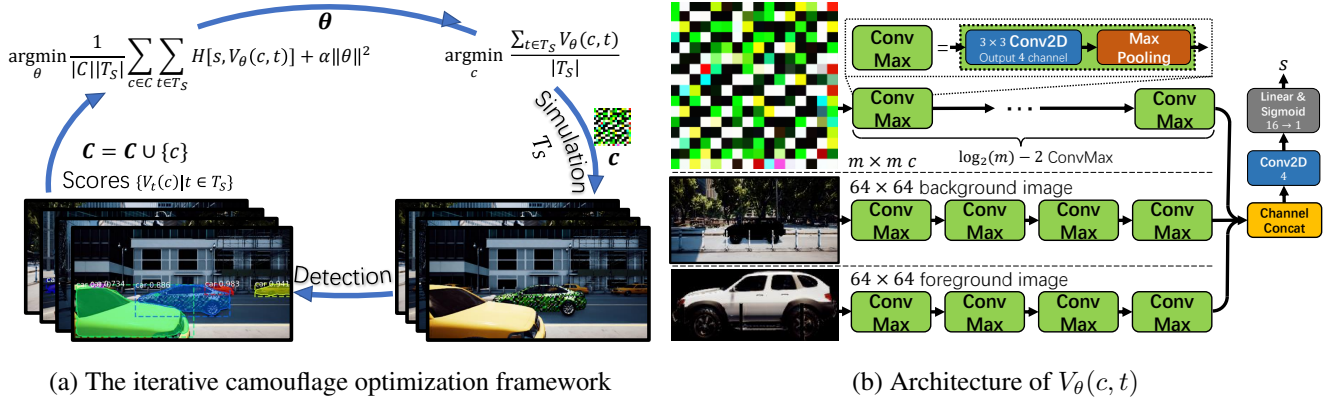


Figure 5.4: Overview of our optimization pipeline and the clone network $V_\theta(c, t)$.

5.3.3 Jointly learning the clone network and the optimal camouflage

We alternatively learn the clone network $V_\theta(c, t)$ and solve the problem (5.2). Once a new camouflage pattern is found due to optimizing problem (5.2), it is converted to multiple images by the training transformations T_S . We obtain the detection score for each of the images by querying a detector. The camouflage pattern, along with the detection scores are then added to the training set for learning the clone network $V_\theta(c, t)$. Fig. 5.4a illustrates this process.

Implementation details. Denote by $H[p, q] := -p \log q - (1 - p) \log(1 - q)$ the cross-entropy loss. We alternately solve the following two problems,

$$\arg \min_{\theta} \frac{1}{|C||T_S|} \sum_{c \in C} \sum_{t \in T_S} H[s, V_\theta(c, t)] + \lambda \|\theta\|_2, \quad \arg \min_c \frac{1}{|T|} \sum_{t \in T_S} H[0, V_\theta(c, t)] \quad (5.3)$$

where C is the collection of all camouflages in the training set for learning the clone network $V_\theta(c, t)$, and $s := V_t(c)$ is the detection score corresponding to the camouflage c and the transformation t . The ℓ_2 regularization $\lambda \|\theta\|_2$ over the clone network's weights is essential. Without this

term, the two loss functions may cause oscillations or degeneration of the solutions. We set $\lambda = 10$ in the experiments. Since the approximation accuracy of the clone network near the optimal camouflage is more important than in other regions, we weigh the newly added samples to the training set 10 times higher than the old ones at each iteration. Algorithm 1 gives more details.

Algorithm 1: Iterative Object Camouflage Learning

Input : Clone network parameter $V_\theta(\cdot)$; Simulation and detection $V_{T_S}(\cdot)$; Transformation T_S which are parameterized as rendered background and foreground images; Regularization tradeoff α ; Random camouflage set C_R .

- 1 Initialize V_θ with random weights θ
 - 2 Set score record $s^* \leftarrow +\infty$
 - 3 $C \leftarrow C_R$
 - 4 **repeat**
 - 5 $\theta \leftarrow \arg \min_\theta \frac{1}{|C||T_S|} \sum_{c \in C} \sum_{t \in T_S} H[s, V_\theta(c, t)] + \lambda \|\theta\|_2$
 - 6 $c' \leftarrow \arg \min_c \frac{1}{|T|} \sum_{t \in T_S} H[0, V_\theta(c, t)]$
 - 7 $s' \leftarrow \{V_t(c) | t \in T\}$
 - 8 $C \leftarrow C \cup \{c'\}$
 - 9 **if** $\text{mean}(s') < s^*$ **then**
 - 10 $s^* \leftarrow \text{mean}(s')$
 - 11 $c^* \leftarrow c'$
 - 12 **until** Reach maximum training steps
- output:** Best learned camouflage c^*
-

5.4 Experiments

Since the primary objective of this chapter is to learn camouflage patterns that deceive vehicle detectors, we introduce two baseline camouflage patterns in the experiments: 6 most popular car colors and 800 random camouflages in different resolutions. We then analyze how the resolution of the camouflage affects the detection performance. For the vehicles, we employ two 3D models: a 2015 Toyota Camry and a virtual SUV. In addition to the main comparison results, we also test the transferability of the learned camouflage across different car models, environments, camera positions, and object detectors.

5.4.1 Experiment setup

We describe the detailed experimental setup in this section, including the simulator, vehicle detector, evaluation metrics, and the baseline camouflage patterns.

5.4.1.1 The simulator



(a) An urban environment.



(b) A mountain environment.

Figure 5.5: The two environments we built to learn and test the vehicle camouflage. Zoom in for more details. These images are of the higher resolution but the same rendering quality (anti-aliasing level, shadow quality, rendering distance, texture resolution etc.) the detector perceived.

As shown in Fig. 5.5a, we use the Unreal engine to build our first simulation environment from the photo-realistic [4] environment. It is modeled after the downtown Manhattan in New York. There are skyscrapers, cars, traffic signs, a park, and roads in this environment, resembling a typical urban environment. We sample 32 different locations along the streets. Eight cameras perch at each location, each taking pictures of the size 720×360 . The relative position of a camera is indexed by the viewing angle and camera-to-object distance, as shown in Fig. 5.3. In addition, we install another 16 cameras to test the generalization of the learned camouflage across viewing angles, distances, etc. In total, we use 18 locations for training and another 18 for testing. Note that the vehicles are invisible to some cameras due to occlusion.

Our second environment is based on a totally different countryside scene called [3], shown in Fig. 5.5b. The roads lie on high-altitude mountains and cross bridges, forest, snow, and a lake. We use this scene to test the transferability of our camouflage across different environments; this scene is not used in the training. Like the [4] environment, we sample 18 locations along the roads for the purpose of testing.

The two vehicles used in the experiments are shown in Fig. 5.7. One is a 2015 Toyota Camry XLE sold in the European Union. The other is a virtual SUV from AirSim [170]. It is worth noting that the Toyota sedan appears multiple times in the MS-COCO dataset [112] (cf. some examples in Fig. 5.6). Since the object detectors are trained on MS-COCO, the Camry is more challenging to hide than the virtual SUV.



Figure 5.6: A fraction of different Toyota sedan appearances in MS COCO dataset. Their appearances are diverse.



Figure 5.7: Orthogonal views of the Toyota Camry XLE 2015 (second row) and the virtual SUV (first row) used in our simulation.

5.4.1.2 Vehicle detectors

We study two state-of-the-art detectors: Mask R-CNN [81] and YOLOv3-SPP [153]. Mask R-CNN is one of the most powerful publicly available object detectors; it currently ranks in the 4th place in the MS COCO detection leaderboard. Both detectors are pre-trained on MS COCO. For Mask R-CNN, we use the one implemented by [5]. Its backbone network is ResNet-101 [82]. YOLOv3 has comparable performance with Mask R-CNN. Its network architecture is very different from Mask R-CNN's, causing challenges to the transfer of the camouflage between the two detectors. We use the spatial pyramid pooling (SPP) variant of YOLOv3 in the experiments.

In the rest of the chapter, we experiment with Mask R-CNN except the transfer experiments in Sec. 5.4.5.

5.4.1.3 Evaluation Metrics

We adopt two metrics to evaluate the detection performance. The first one is a variation of the Intersection over Union proposed by [49]. The IoU between a predicted box and the groundtruth bounding box is defined as $IoU(A, B) = \frac{A \cap B}{A \cup B}$. Since IoU was originally proposed to evaluate the results of multi-object detection, as opposed to the single vehicle detection in our context, we modify it to the following to better capture the vehicle of interest: $\max_{p \in P} IoU(p, GT)$, where P is the set of all detection proposals in an image. We average this quantity across all the test images and denote by mIoU the mean value.

Our second metric is precision at 0.5 or P@0.5. [49] set a 0.5 threshold for the detection IoU in the PASCAL VOC detection challenge to determine whether a detection is a hit or miss. We report the percentage of the hit detections out of all observations as our P@0.5. We also report the relative precision drop of the camouflage against the baseline colors whenever possible.

5.4.1.4 Baselines

Our first baseline is when a vehicle is colored in popular real car colors. We select 6 basic car colors (red, black, silver, grey, blue, and white) which cover over 90% of the car colors worldwide [14]. We obtain their RGB values according to the [1].

As the second baseline, we generate 800 random camouflages in different resolutions ranging in $\{2^i \times 2^i; i \in [1..8]\}$. Since we find that camouflages with strong contrasts work better, we generate half of the camouflages using RGB values $\in [0, 255]$ and the other half using RGB values $\in \{0, 255\}$. After we obtain these camouflages and the corresponding detection scores, we use those with proper resolutions to initialize the training set for the clone network $V_\theta(c, t)$.

The two baselines partially resolve the concern one might have that the learned camouflage successfully attacks the detector not because it exploits the CNN’s structural weakness, but because it takes advantage of the domain gap between the real data used to train the detector and our simulated data used to test the detector. Results show that the baselines could not fail the detectors under most test cases, indicating that the detectors are fairly resilient to the domain gap between the simulated imagery and the real data (at least for the vehicle detection task).

5.4.2 Resolution of the camouflages

We first report the random camouflages’ performance on hiding the Camry from the Mast R-CNN detector in the DownTown environment. Fig. 5.8 shows the results. The first observation is that, although random camouflages are visually very different from conventional car paintings (cf. Fig. 5.9), the Mask R-CNN detector is still able to detect most of them. Another slightly counter-intuitive yet interesting observation is that the detector’s performance does not always decrease as the camouflages’ resolutions increase. This is probably because some fine-grained patterns

displayed by the high-resolution camouflage become harder to observe from a distance. Since the high-resolution camouflage does not bring any additional benefits beyond the 16×16 resolution, we use camouflages of size 16×16 in the remaining experiments.

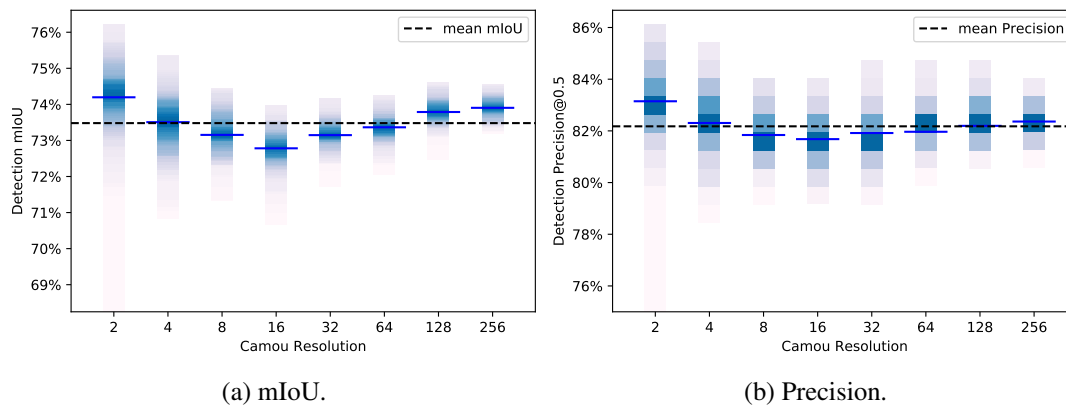


Figure 5.8: The mIoU and P@0.5 of the 800 random camouflages in different resolutions on Camry in the DownTown environment. There are 100 camouflages per resolution.



Figure 5.9: Visualization of three random camouflages of resolutions 4×4 , 16×16 , and 256×256 , respectively, as well as the resulting images by the same camera.

5.4.3 Camouflaging Toyota Camry in the urban environment

Here we report the results on detecting the Camry in the urban environment. Table 5.1 summarizes the results for the Camry respectively with the baseline colors, random camouflages, and our

learned camouflage. We can see from the table that the Mask R-CNN is surprisingly robust against different types of random camouflages. The random camouflages’ detection scores are close to the baseline colors’. Moreover, the standard deviation of the random camouflages’ scores is also very low, indicating that the difference in the random camouflages does not really change the detector’s performance. Note that we apply the camouflage to the body of the Camry, leaving tires, windows, grilles, etc. as informative visual cues to the detector. Despite that, our learned camouflage reduces the precision by around 30% over baseline colors in both training and testing scenes in the urban environment.

Table 5.1: Mask R-CNN detection performance on the Camry in the urban environment.

Camouflages	Training Scenes		Testing Scenes	
	mIoU (%)	P@0.5 (%)	mIoU (%)	P@0.5 (%)
Baseline Colors	76.14	84.40	72.88	77.57
Random Camou	73.48± 0.80	82.17± 1.20	67.79± 0.79	71.42± 1.20
Ours	57.69	62.14	53.64	52.17
Relative Performance Drop	24.23%	26.37%	26.39%	32.74%

5.4.4 Virtual SUV in urban area

We then report the camouflage performance on the newly modeled SUV in the urban environment in Table 5.2.

Table 5.2: Detection performance of camouflages on SUV in urban environment.

Camouflages	Training Scenes		Testing Scenes	
	mIoU (%)	P@0.5 (%)	mIoU (%)	P@0.5 (%)
Baseline Colors	82.35	91.07	81.06	89.22
Random Camou	83.53±3.26	93.21±3.48	78.02±2.53	84.79±2.81
Ours	53.27	55.79	48.91	50.36
Relative Performance Drop	35.31%	38.79%	39.66%	43.55%

Judging by the results, it is clear that the Mask R-CNN is a very generalized detector. The SUV’s baseline color and random camouflage detection scores are even higher than the Camry correspondence although the detector has never seen it before. This might be because the SUV has much less polygon and is built to resemble the shape of a general SUV as shown in Fig. 5.7.

However, the price for not seeing this vehicle during training is that Mask R-CNN is more likely to be affected by the camouflage. The standard deviation of the random camouflage mIoU/ precision is higher (3.26/3.48 vs. 0.8/1.2) than Camry. And our camouflage achieves lower detection precision despite both baseline colors and random camouflage’s detection scores are higher than Camry’s. The detectability is reduced by almost 1.5 times of Camry’s results in this case. This experiment shows that Mask R-CNN might well generalize to unseen vehicles, but it is easier to get attacked.

5.4.5 Transferability across detectors

We show that the camouflage learned to attack Mask R-CNN can actually also defeat YOLOv3 to a certain degree. The results are reported in Table 5.3.

Table 5.3: YOLOv3-SPP detection performance on the Camry in the urban environment. In addition to the camouflage inferred for YOLO, we also include the YOLO detection results on the camouflage learned for Mask R-CNN.

Camouflages	Training Scenes		Testing Scenes	
	mIoU (%)	P@0.5 (%)	mIoU (%)	P@0.5 (%)
Baseline colors	76.79	85.34	73.00	78.50
Random Camou	73.19±0.75	83.30±0.89	68.76±0.81	73.92±1.00
Ours (YOLO trained)	65.83	72.53	64.03	69.56
Ours (Mask R-CNN trained)	65.43	70.42	61.79	65.21
Relative Performance Drop	14.79%	17.48%	15.35%	16.92%

5.4.6 Transferability across environments

Another important question is that what if we transfer the camouflage to a not only previously unseen but a totally different environment? To quantitatively answer this question, we build the Landscape environment (Fig. 5.5b) to test our camouflages trained in urban environment (Fig. 5.5a) using the Camry vehicle. The results are reported in Table. 5.4. Some qualitative results are shown in Fig. 5.12.

Table 5.4: Detection performance of camouflages on Camry in Landscape environment. Note that this camouflage is pretrained in urban environment and then transferred without any finetuning.

Camouflages	Testing Scenes	
	mIoU (%)	P@0.5 (%)
Baseline Colors	74.81	82.04
Random Camou	72.45±4.26	77.11±5.45
Ours - Transferred	40.39	43.26
Relative Performance Drop	46.00%	47.26%

Given the barren landscape with few objects in sight, the detector detects the car better than it did in the urban environment for both baseline colors (82.04 vs. 77.57) and random camouflages (77.11 vs. 71.42) possibly due to the absence of distractions. However, our directly transferred camouflage is still able to beat both of them by more than 46% regarding both detection mIoU and precision without any fine-tuning.

It is interesting to notice that the Camry with grey color looks almost identical to the background in this environment (Fig. 5.12). However, it still could be perfectly detected. Meanwhile, our leaned camouflage results in far better stealth despite it has a sharp contrast to the background.

5.4.7 Transferability across vehicles

It will be impractical to retrain a specific camouflage for each vehicle whenever we need it. Hence it would be interesting to look into the transferability between vehicles in this scenario. We swap the camouflages of SUV and Camry and see how they would perform in the urban environment. We present the testing precision after swapping in Table. 5.5.

Table 5.5: Camouflage transferability across vehicle reported in testing P@0.5 in urban environment.

		Test	
		SUV	Camry
Train	SUV	50.36	47.44
	Camry	58.39	52.17

First, both camouflages are definitely transferable. It is also interesting to see that the Camry learned camouflage is not as good as the SUV learned camouflage even when being applied on the Camry itself. This might be due to the fact that the SUV resembles a car with more generic features and hence learning camouflage on SUV is less likely to encounter local minima during optimization.

5.4.8 Transferability across viewing positions

One of the possibly most concerning questions is whether the learned camouflage is robust to the change of the camera position. To answer this question, we set up another 16 new cameras, as shown in Fig. 5.3, surrounding the vehicle in different relative locations. We then test the learned camouflage’s performance on these new cameras. The results are shown in Table 5.6.

Table 5.6: Detection performance of pretrained camouflages on Camry with urban environment in 16 unseen cameras.

Camouflages	Testing Scenes	
	mIoU (%)	P@0.5 (%)
Baseline Colors	78.43	86.16
Random Camou	77.26±0.96	84.61±1.06
Ours - Transferred	67.74	73.14
Relative Performance Drop	13.62%	15.11%

Our camouflage performance drop has a slightly decrease of 5% from Table 5.1. This indicates that the change of the camera locations would impact the performance, but the performance drop is still way beyond the standard deviation of random camouflages’ scores. Given that the new camera views cover more perspectives as shown in Fig. 5.3, this result is reasonable.

5.4.9 Impact of clone network quality

How does the clone network’s quality affect the camouflage’s performance? Is the alternative optimization necessary? We quantitatively show the first 300 simulation calls of our system in Fig. 5.10 and meanwhile evaluate the camouflages proposed by the clone network. Note that initially the clone network has already been trained with 800 random camouflages. However, the proposed camouflage’s score does not fall until the new camouflages from iteration scheme join the optimization. This suggests that without our iterative optimization mechanism, the clone network could only find camouflage with mIoU around 70%, which is the same as the random camouflage. Those new samples serve as the hard samples. They gradually calibrate the clone network’s global minima to $V_t()$ ’s and help it to generate better camouflages. Note that although the score descends

quicker during the first 50 iterations, it does not find the global best camouflage until near the end (296th iteration). This graph also shows the classification score is a suitable choice to be minimized as it is highly correlated with mIoU.

Note that the system automatically re-initialize the clone network's parameters to prevent it from falling into local minima during the optimization whenever we add a new sample to the training set C . Hence, there are some spikes in the graph.

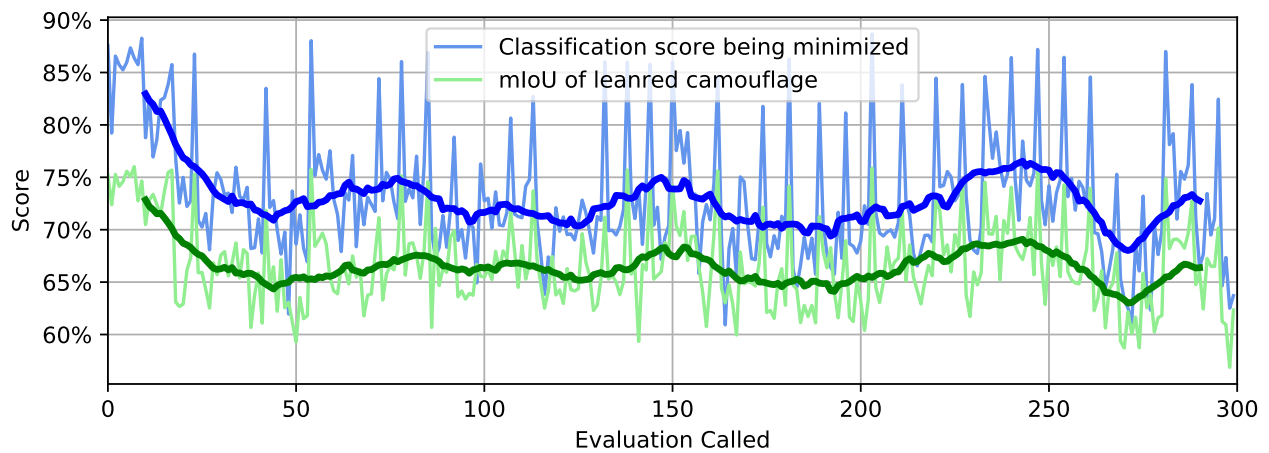


Figure 5.10: Clone network's learned camouflage's classification score and mIoU vs. Simulation called. We can see how the new samples helped the clone network to find the minimal.

5.4.10 Detection attention

How exactly does our camouflage work against the detector? We partially answer this question by visualizing Mask-RCNN's completely/partially successful detection attention on our grey and camouflaged Camry. Since our Mask-RCNN implementation does not explicitly yield failed detection's prediction, we are unable to visualize the failed detection's attention w.r.t. the input image if the failed detection does not exist.

There are two main visualization approaches: Grad-CAM [168] and saliency [174]. Approximately speaking, grad-CAM visualizes the gradient of output w.r.t. penultimate (pre-fully-connected layer) convolutional layer feature map output; Saliency visualizes the gradient of output w.r.t. initial input image. Grad-CAM is generally considered superior as the last convolutional layer’s feature map contains much more abstracted semantic information, leading to less noisy visualization. However, we find that it is hard to define the single “penultimate layer” in Mask-RCNN: It has multiple penultimate layers, tracing back to different stages of the network, prior to the ROI pooling layer. Each of those penultimate layers contains varying levels of information. We choose to use saliency in this case.

It is clear how to define the “attention” in the image classification scenario: It is the gradient heatmap of the classification score scalar w.r.t. the entire input image. On the other hand, an end-to-end detection neural network often yields multiple structure predictions. Each structure contains bounding box and classification score, etc. We choose to visualize the gradient of the best bounding-box’s classification score w.r.t. the input image.

Our visualizations are presented in Fig. 5.11. It is surprising that the window, roof and upper bodies are playing the predominant role in car detection. Upper body attention exists even when the upper body is not included in the detection bounding box (row 3). Given that the classification stage makes the classification decision based on the proposed feature map box region, such attention must have been already included in the detection proposal before the ROI layer, i.e. detection stage. This may also explain why it is easier to fail the front-viewing and the rear-view detectors: front-view (row 1) and the rear-view (row 5) detectors place their attention on the hood and trunk, where camouflage pattern is presented. A partially successful attack (row 4) was carried out by wiping out detector’s attention on the car hood. On the other hand, the side-view detector is harder to attack since the detector merely places any attention of the car body (row 2) where the camouflages are mainly located. However, our camouflages on the roof could still fail the side-

view detector partially (row 3). Since we only visualize the (partially) successful detections, there are still many cases to explore.



Figure 5.11: The best detections in each image and the gradient heatmap of their classification scores w.r.t. the input images. The detector places its attention predominantly on the upper car body, i.e., roof, hood, trunk, and windows.

5.4.11 Simulation implementation

The iterative optimization framework works on two Nvidia GTX 1080 Ti in our experiment. We implement the framework using PyTorch [142]. We use one to run the detector and another one to

run the simulation and training/ prediction of evaluation network. The simulation was implemented partially using AirSim by [170] and UnrealEnginePython. All submodules are implemented asynchronously to run in parallel and are communicating with each other using RPC/ RPyC. All the camouflages and the baseline colors are implemented either using or based on the official Unreal Automotive Material. Each evaluation in the simulation, which is the most time-consuming part, takes around 15 to 20 second.

5.4.12 Simulation error

Despite our best effort, we observe $V_t(c)$ come with a standard deviation of 0.008 due to the inherent and mostly necessary random processes in the rendering (i.e., Monte Carlo in path tracing, etc.). This unfortunately makes $V_t(c)$ a noisy function. We reduce this error by repeat sampling $V_t(c)$ 5 times whenever we use it.

5.4.13 Non-linearity and non-convexity

Since this is a blackbox optimization problem, it is important to examine some important features of the $\mathbb{E}_t V_t(\cdot)$. We first verify its convexity via the convexity definition. We test the convexity of $\mathbb{E}_t V_t(\cdot)$ by testing the convexity of subsampled correspondence $\frac{1}{|T_S|} \sum_{t \in T_S} V_t(\cdot)$ via:

$$\forall c_1, c_2 \in C : \quad \frac{1}{|T_S|} \sum_{t \in T_S} V_t\left(\frac{c_1 + c_2}{2}\right) \leq \frac{1}{|T_S|} \sum_{t \in T_S} V_t \frac{V_{T_S}(c_1) + V_{T_S}(c_2)}{2} \quad (5.4)$$

where C is a set of camouflages. We sampled 1000 pairs of random camouflages from C and half of them do not meet the equation. Hence $\frac{1}{|T_S|} \sum_{t \in T_S} V_t(\cdot)$ is nonconvex.

Besides, we find a simple linear MLP is insufficient to approximate $\frac{1}{|T_S|} \sum_{t \in T_S} V_t(\cdot)$, which empir-

ically shows it is nonlinear.

5.4.14 Qualitative results



Figure 5.12: Qualitative comparison of the Mask R-CNN detections results of the grey baseline color, random camouflages and our learned camouflages in different transformations. Zoom in for more details.

We present some of our detection results on the baseline colors, random camouflages, and the learned camouflages for the Camry in Fig. 5.12.

We can draw a lot of interesting observations from the qualitative results which were hidden by the quantitative results. We find that there are 3 types of successful attacks: (1) Camouflages lower the objectiveness of the car and the car region is not proposed or only partially proposed as a candidate for the classifier of the detector; (2) The car region is successfully proposed but misclassified (e.g., to kite, cake, truck, or potted plant as shown in the examples) or the classification score is too low to pass the threshold for the detection score; (3) The car region is successfully proposed and classified, but the camouflage results in an incorrect detection which largely overlaps with the car (cf. the 5th row where the regions covering the car are detected as a car and a boat, respectively).

One can see that the context or background plays a vital role in object detection. Although some images capture the Camry by the same pose, the detector makes completely different predictions for them. Besides, these qualitative results imply that the detector works in a way different from human vision. In the Landscape environment, our learned camouflage has the strongest contrast to the background compared to other baseline patterns. However, the detector is still sometimes not able to detect the camouflaged car or SUV.

5.5 Summary

In this chapter, we investigate whether it is possible to physically camouflage 3D objects of complex shapes, i.e., vehicles, in order to hide them from state-of-the-art object detectors. We conduct extensive experimental studies with a photo-realistic simulation engine. We propose to use a clone network to mimic the simulator and the detector’s joint response to the 3D vehicles. Then, we infer a camouflage for a 3D vehicle by minimizing the output of the clone network. Our learned cam-

ouflage significantly reduces the detectability of a Toyota Camry and a SUV. Moreover, we find that the camouflage is transferable across different environments. For future work, We plan to look into possible ways to white-box the entire process so as to propose a more effective camouflage.

CHAPTER 6: CONCLUSION

Deep learning has offered many challenges as well as opportunities to the transfer learning. In this paper, we look into the problems of transfer learning in the context of deep learning. Specifically, we listed three unique problems: zero-shot image tagging, domain adaptation for segmentation and learning transferable camouflage. For zero-shot image tagging in chapter 3, we assume the existence of principle direction for a given image, where relevant tags rank ahead of the irrelevant tag along principle direction. Then we solve the tagging problem by finding the mapping between images and principle directions. We propose curriculum-style domain adaptation for domain adaptation for segmentation in chapter 4. This is done by inferring target properties first. Then we use target properties to improve the training of segmentation network on target distribution. As for generalizable adversarial camouflage, we propose to combine simulation with expectation over transformation to learn a transformation-insensitive, thus generalizable, camouflage in chapter 5.

LIST OF REFERENCES

- [1] X11 Color Names. tex.key: X11 Color Names.
- [2] UnrealEngine, 1998. tex.key: Unreal.
- [3] Landscape Enviroment, 2014. tex.key: Landscape Mountains.
- [4] Downtown Enviroment, 2017. tex.key: DownTown.
- [5] W. Abdulla. Mask R-CNN for object detection and instance segmentation on Keras and TensorFlow. *GitHub repository*, 2017. tex.publisher: Github.
- [6] Z. Akata, F. Perronnin, Z. Harchaoui, and C. Schmid. Label-Embedding for Attribute-Based Classification. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 819–826, June 2013.
- [7] Z. Akata, S. Reed, D. Walter, H. Lee, and B. Schiele. Evaluation of Output Embeddings for Fine-Grained Image Classification. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2927–2936, 2015.
- [8] N. Akhtar and A. Mian. Threat of adversarial attacks on deep learning in computer vision: A survey. *arXiv preprint arXiv:1801.00553*, 2018.
- [9] R. Al-Rfou, G. Alain, A. Almahairi, C. Angermueller, D. Bahdanau, N. Ballas, F. Bastien, J. Bayer, A. Belikov, A. Belopolsky, and others. Theano: A Python framework for fast computation of mathematical expressions. *arXiv preprint arXiv:1605.02688*, 2016.
- [10] R. Aljundi, R. Emonet, D. Muselet, and M. Sebban. Landmarks-based kernelized subspace alignment for unsupervised domain adaptation. In *IEEE conference on computer vision and pattern recognition (CVPR)*, pages 56–63, 2015.
- [11] A. Arnab, O. Miksik, and P. H. S. Torr. On the robustness of semantic segmentation models to adversarial attacks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018.

- [12] A. Athalye, N. Carlini, and D. Wagner. Obfuscated gradients give a false sense of security: Circumventing defenses to adversarial examples. In *Proceedings of the 35th international conference on machine learning*, July 2018.
- [13] A. Athalye, L. Engstrom, A. Ilyas, and K. Kwok. Synthesizing robust adversarial examples. In *Proceedings of the 35th international conference on machine learning*, volume 80, pages 284–293, 2018.
- [14] Axalta. Global automotive 2017 color popularity report. 2017. tex.key: Axalta.
- [15] V. Badrinarayanan, A. Kendall, and R. Cipolla. Segnet: A deep convolutional encoder-decoder architecture for image segmentation. *Transactions of Pattern Recognition and Machine Analyses (PAMI)*, 2016.
- [16] C. Barata, M. E. Celebi, and J. S. Marques. Improving dermoscopy image classification using color constancy. *IEEE journal of biomedical and health informatics*, 19(3):1146–1152, 2015. tex.publisher: IEEE.
- [17] K. Barnard, P. Duygulu, D. Forsyth, N. De Freitas, D. M. Blei, and M. I. Jordan. Matching words and pictures. *The Journal of Machine Learning Research*, 3:1107–1135, 2003.
- [18] Y. Bengio, J. Louradour, R. Collobert, and J. Weston. Curriculum learning. In *International conference on machine learning (ICML)*, pages 41–48, 2009. tex.organization: ACM.
- [19] J. Bergstra, O. Breuleux, F. Bastien, P. Lamblin, R. Pascanu, G. Desjardins, J. Turian, D. Warde-Farley, and Y. Bengio. Theano: a CPU and GPU math expression compiler. In *Proceedings of the Python for scientific computing conference (SciPy)*, volume 4, page 3. Austin, TX, 2010.
- [20] H. Bilen, M. Pedersoli, and T. Tuytelaars. Weakly supervised object detection with posterior regularization. In *BMVA british machine vision conference (BMVC)*, 2014.
- [21] K. Bousmalis, A. Irpan, P. Wohlhart, Y. Bai, M. Kelcey, M. Kalakrishnan, L. Downs, J. Ibarz, P. Pastor, K. Konolige, and others. Using simulation and domain adaptation to improve efficiency of deep robotic grasping. *arXiv preprint arXiv:1709.07857*, 2017.

- [22] K. Bousmalis, N. Silberman, D. Dohan, D. Erhan, and D. Krishnan. Unsupervised pixel-level domain adaptation with generative adversarial networks. *arXiv preprint arXiv:1612.05424*, 2016.
- [23] G. J. Brostow, J. Fauqueur, and R. Cipolla. Semantic object classes in video: A high-definition ground truth database. *Pattern Recognition Letters*, 30(2):88–97, 2009.
- [24] G. J. Brostow, J. Shotton, J. Fauqueur, and R. Cipolla. Segmentation and recognition using structure from motion point clouds. In *European conference on computer vision (ECCV)*, pages 44–57, 2008.
- [25] T. B. Brown, D. Man, A. Roy, M. Abadi, and J. Gilmer. Adversarial patch. *arXiv preprint arXiv:1712.09665*, 2017.
- [26] C. Bucilu, R. Caruana, and A. Niculescu-Mizil. Model compression. In *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 535–541, 2006. tex.organization: ACM.
- [27] C. Burges, T. Shaked, E. Renshaw, A. Lazier, M. Deeds, N. Hamilton, and G. Hullender. Learning to rank using gradient descent. In *Proceedings of the 22nd international conference on Machine learning*, pages 89–96. ACM, 2005.
- [28] N. Carlini and D. Wagner. Audio adversarial examples: Targeted attacks on speech-to-text. *arXiv preprint arXiv:1801.01944*, 2018.
- [29] G. Carneiro, A. B. Chan, P. J. Moreno, and N. Vasconcelos. Supervised learning of semantic classes for image annotation and retrieval. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 29(3):394–410, 2007.
- [30] O. Chapelle and S. S. Keerthi. Efficient algorithms for ranking with SVMs. *Information Retrieval*, 13(3):201–215, 2010.
- [31] H. Chen, H. Zhang, P.-Y. Chen, J. Yi, and C.-J. Hsieh. Attacking visual language grounding with adversarial examples: A case study on neural image captioning. In *Proceedings of the 56th annual meeting of the association for computational linguistics*, volume 1, pages 2587–2597, 2018.
- [32] L.-C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille. Semantic image segmentation with deep convolutional nets and fully connected crfs. *arXiv preprint arXiv:1412.7062*, 2014.

- [33] L.-C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille. Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. *Transactions of Pattern Recognition and Machine Analyses (PAMI)*, 40(4):834–848, 2018. tex.publisher: IEEE.
- [34] M. Chen, A. Zheng, and K. Weinberger. Fast image tagging. In *Proceedings of the 30th international conference on Machine Learning*, pages 1274–1282, 2013.
- [35] P.-Y. Chen, H. Zhang, Y. Sharma, J. Yi, and C.-J. Hsieh. Zoo: Zeroth order optimization based black-box attacks to deep neural networks without training substitute models. In *Proceedings of the 10th ACM workshop on artificial intelligence and security*, pages 15–26, 2017. tex.organization: ACM.
- [36] S.-T. Chen, C. Cornelius, J. Martin, and D. H. Chau. ShapeShifter: Robust physical adversarial attack on faster r-cnn object detector. *arXiv preprint arXiv:1804.05810*, 2018.
- [37] Y. Chen, W. Li, and L. Van Gool. ROAD: Reality oriented adaptation for semantic segmentation of urban scenes. *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.
- [38] Y.-H. Chen, W.-Y. Chen, Y.-T. Chen, B.-C. Tsai, Y.-C. F. Wang, and M. Sun. No more discrimination: Cross city adaptation of road scene segmenters. In *IEEE international conference on computer vision (ICCV)*, pages 2011–2020, 2017. tex.organization: IEEE.
- [39] T.-S. Chua, J. Tang, R. Hong, H. Li, Z. Luo, and Y. Zheng. NUS-WIDE: a real-world web image database from National University of Singapore. In *Proceedings of the ACM international conference on image and video retrieval*, page 48. ACM, 2009.
- [40] M. Cordts, M. Omran, S. Ramos, T. Rehfeld, M. Enzweiler, R. Benenson, U. Franke, S. Roth, and B. Schiele. The cityscapes dataset for semantic urban scene understanding. In *IEEE conference on computer vision and pattern recognition (CVPR)*, pages 3213–3223, 2016. tex.date-added: 2017-03-14 21:26:33 +0000 tex.date-modified: 2017-03-14 21:26:33 +0000.
- [41] C. Cortes and V. Vapnik. Support-vector networks. *Machine learning*, 20(3):273–297, 1995.
- [42] K. Crammer and Y. Singer. On the algorithmic implementation of multiclass kernel-based vector machines. *The Journal of Machine Learning Research*, 2:265–292, 2002.

- [43] G. Csurka. Domain adaptation for visual applications: A comprehensive survey. *arXiv preprint arXiv:1702.05374*, 2017.
- [44] J. Dai, K. He, and J. Sun. Instance-aware semantic segmentation via multi-task network cascades. In *IEEE conference on computer vision and pattern recognition (CVPR)*, pages 3150–3158, 2016.
- [45] S. C. Deerwester, S. T. Dumais, T. K. Landauer, G. W. Furnas, and R. A. Harshman. Indexing by latent semantic analysis. *JAsIs*, 41(6):391–407, 1990.
- [46] A. Dehghan, H. Idrees, and M. Shah. Improving Semantic Concept Detection through the Dictionary of Visually-distinct Elements. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2585–2592, 2014.
- [47] A. Dosovitskiy, G. Ros, F. Codevilla, A. Lopez, and V. Koltun. CARLA: An open urban driving simulator. In *Proceedings of the 1st annual conference on robot learning*, pages 1–16, 2017.
- [48] M. Everingham, S. A. Eslami, L. Van Gool, C. K. Williams, J. Winn, and A. Zisserman. The pascal visual object classes challenge: A retrospective. *International Journal of Computer Vision (IJCV)*, 111(1):98–136, 2015.
- [49] M. Everingham, S. M. A. Eslami, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The pascal visual object classes challenge: A retrospective. *International Journal of Computer Vision*, 111(1):98–136, Jan. 2015.
- [50] K. Eykholt, I. Evtimov, E. Fernandes, B. Li, A. Rahmati, F. Tramer, A. Prakash, T. Kohno, and D. Song. Physical adversarial examples for object detectors. In *12th USENIX workshop on offensive technologies (WOOT 18)*, Baltimore, MD, 2018. USENIX Association.
- [51] K. Eykholt, I. Evtimov, E. Fernandes, B. Li, A. Rahmati, C. Xiao, A. Prakash, T. Kohno, and D. Song. Robust physical-world attacks on deep learning visual classification. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1625–1634, 2018.
- [52] A. Farhadi, I. Endres, D. Hoiem, and D. Forsyth. Describing objects by their attributes. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1778–1785. IEEE, 2009.

- [53] P. Felzenszwalb, D. McAllester, and D. Ramanan. A discriminatively trained, multiscale, deformable part model. In *2008 IEEE conference on computer vision and pattern recognition*, pages 1–8, 2008. tex.organization: IEEE.
- [54] S. Feng, R. Manmatha, and V. Lavrenko. Multiple bernoulli relevance models for image and video annotation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, volume 2, pages II–1002. IEEE, 2004.
- [55] B. Fernando, A. Habrard, M. Sebban, and T. Tuytelaars. Unsupervised visual domain adaptation using subspace alignment. In *IEEE international conference on computer vision (ICCV)*, pages 2960–2967, 2013.
- [56] D. H. Foster. Color constancy. *Vision research*, 51(7):674–700, 2011. tex.publisher: Elsevier.
- [57] Francois Chollet. keras. *GitHub repository*, 2015. tex.commit: 5bcac37 tex.publisher: GitHub.
- [58] A. Frome, G. S. Corrado, J. Shlens, S. Bengio, J. Dean, T. Mikolov, and others. Devise: A deep visual-semantic embedding model. In *Advances in Neural Information Processing Systems*, pages 2121–2129, 2013.
- [59] J. Fu, Y. Wu, T. Mei, J. Wang, H. Lu, and Y. Rui. Relaxing From Vocabulary: Robust Weakly-Supervised Deep Learning for Vocabulary-Free Image Tagging. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1985–1993, 2015.
- [60] Y. Fu, Y. Yang, T. Hospedales, T. Xiang, and S. Gong. Transductive Multi-label Zero-shot Learning. pages 7.1–7.11. British Machine Vision Association, 2014.
- [61] Y. Fu, Y. Yang, T. M. Hospedales, T. Xiang, and S. Gong. Transductive Multi-class and Multi-label Zero-shot Learning. *arXiv preprint arXiv:1503.07884*, 2015.
- [62] A. Gaidon, Q. Wang, Y. Cabon, and E. Vig. Virtual worlds as proxy for multi-object tracking analysis. In *IEEE conference on computer vision and pattern recognition (CVPR)*, pages 4340–4349, 2016.
- [63] A. Gaidon, Q. Wang, Y. Cabon, and E. Vig. Virtual worlds as proxy for multi-object tracking analysis. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4340–4349, 2016.

- [64] K. Ganchev, J. Gillenwater, B. Taskar, and others. Posterior regularization for structured latent variable models. *Journal of Machine Learning Research*, 11(Jul):2001–2049, 2010.
- [65] Y. Ganin and V. Lempitsky. Unsupervised domain adaptation by backpropagation. In *International conference on machine learning (ICML)*, pages 1180–1189, 2015.
- [66] Y. Ganin, E. Ustinova, H. Ajakan, P. Germain, H. Larochelle, F. Laviolette, M. Marchand, and V. S. Lempitsky. Domain-adversarial training of neural networks. *CoRR*, arXiv:1505.07818, 2015.
- [67] A. Garcia-Garcia, S. Orts-Escolano, S. Oprea, V. Villena-Martinez, and J. Garcia-Rodriguez. A review on deep learning techniques applied to semantic segmentation. *arXiv preprint arXiv:1704.06857*, 2017.
- [68] A. Geiger, P. Lenz, C. Stiller, and R. Urtasun. Vision meets robotics: The KITTI dataset. *The International Journal of Robotics Research*, 32(11):1231–1237, 2013. tex.date-added: 2017-03-14 21:28:35 +0000 tex.date-modified: 2017-03-14 21:28:35 +0000 tex.publisher: Sage Publications Sage UK: London, England.
- [69] A. Geiger, P. Lenz, and R. Urtasun. Are we ready for autonomous driving? The KITTI vision benchmark suite. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2012.
- [70] A. Gijsenij, T. Gevers, and J. Van De Weijer. Generalized gamut mapping using image derivative structures for color constancy. *International Journal of Computer Vision (IJCV)*, 86(2-3):127–139, 2010. tex.publisher: Springer.
- [71] A. Gijsenij, T. Gevers, and J. Van De Weijer. Computational color constancy: Survey and experiments. *IEEE Transactions on Image Processing*, 20(9):2475–2489, 2011. tex.publisher: IEEE.
- [72] B. Gong, K. Grauman, and F. Sha. Connecting the dots with landmarks: Discriminatively learning domain invariant features for unsupervised domain adaptation. In *International conference on machine learning (ICML)*, pages 222–230, 2013.
- [73] B. Gong, F. Sha, and K. Grauman. Overcoming dataset bias: An unsupervised domain adaptation approach. In *NIPS workshop on large scale visual recognition and retrieval (LSVRR)*, 2012.

- [74] B. Gong, Y. Shi, F. Sha, and K. Grauman. Geodesic flow kernel for unsupervised domain adaptation. In *IEEE conference on computer vision and pattern recognition (CVPR)*, pages 2066–2073, 2012. tex.organization: IEEE.
- [75] Y. Gong, Y. Jia, T. Leung, A. Toshev, and S. Ioffe. Deep convolutional ranking for multilabel image annotation. *arXiv preprint arXiv:1312.4894*, 2013.
- [76] R. Gopalan, R. Li, and R. Chellappa. Domain adaptation for object recognition: An unsupervised approach. In *IEEE international conference on computer vision (ICCV)*, pages 999–1006, 2011.
- [77] A. Gretton, K. M. Borgwardt, M. J. Rasch, B. Schölkopf, and A. Smola. A kernel two-sample test. *Journal of Machine Learning Research*, 13(Mar):723–773, 2012.
- [78] A. Gretton, A. Smola, J. Huang, M. Schmittfull, K. Borgwardt, and B. Schölkopf. Covariate shift by kernel mean matching. In J. Quionero-Candela, M. Sugiyama, A. Schwaighofer, and N. D. Lawrence, editors, *Dataset shift in machine learning*. The MIT Press, 2008.
- [79] M. Grubinger, P. Clough, H. Müller, and T. Deselaers. The iapr tc-12 benchmark: A new evaluation resource for visual information systems. In *International Workshop OntoImage*, pages 13–23, 2006.
- [80] M. Guillaumin, T. Mensink, J. Verbeek, and C. Schmid. Tagprop: Discriminative metric learning in nearest neighbor models for image auto-annotation. In *Computer Vision, 2009 IEEE 12th International Conference on*, pages 309–316. IEEE, 2009.
- [81] K. He, G. Gkioxari, P. Dollr, and R. Girshick. Mask r-cnn. In *International conference on computer vision*, pages 2980–2988, 2017. tex.organization: IEEE.
- [82] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.
- [83] G. Hinton, O. Vinyals, and J. Dean. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015. tex.date-added: 2017-03-14 22:02:04 +0000 tex.date-modified: 2017-03-14 22:02:04 +0000.
- [84] G. E. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever, and R. R. Salakhutdinov. Improving neural networks by preventing co-adaptation of feature detectors. *arXiv preprint arXiv:1207.0580*, 2012.

- [85] J. Hoffman, E. Tzeng, T. Park, J.-Y. Zhu, P. Isola, K. Saenko, A. A. Efros, and T. Darrell. CyCADA: Cycle-consistent adversarial domain adaptation. *International Conference on Machine Learning (ICML)*, 2018.
- [86] J. Hoffman, D. Wang, F. Yu, and T. Darrell. FCNs in the Wild: Pixel-level Adversarial and Constraint-based Adaptation. *arXiv preprint arXiv:1612.02649*, 2016.
- [87] S. Hong, J. Oh, H. Lee, and B. Han. Learning transferrable knowledge for semantic segmentation with deep convolutional neural network. In *IEEE conference on computer vision and pattern recognition (CVPR)*, pages 3204–3212, 2016.
- [88] W. Hong, Z. Wang, M. Yang, and J. Yuan. Conditional generative adversarial network for structured domain adaptation. In *IEEE conference on computer vision and pattern recognition (CVPR)*, pages 1335–1344, 2018.
- [89] Z. Hu, X. Ma, Z. Liu, E. Hovy, and E. Xing. Harnessing deep neural networks with logic rules. *arXiv preprint arXiv:1603.06318*, 2016.
- [90] H. Huang, Q. Huang, and P. Krhenbhl. Domain transfer through deep activation matching. In *European conference on computer vision (ECCV)*, pages 611–626, 2018. tex.organization: Springer.
- [91] B. Huval, T. Wang, S. Tandon, J. Kiske, W. Song, J. Pazhayampallil, M. Andriluka, P. Rajpurkar, T. Migimatsu, R. Cheng-Yue, F. Mujica, A. Coates, and A. Y. Ng. An empirical evaluation of deep learning on highway driving. *arXiv preprint arXiv:1504.01716*, 2015.
- [92] D. Jayaraman and K. Grauman. Zero-shot recognition with unreliable attributes. In *Advances in Neural Information Processing Systems*, pages 3464–3472, 2014.
- [93] J. Jeon, V. Lavrenko, and R. Manmatha. Automatic image annotation and retrieval using cross-media relevance models. In *Proceedings of the 26th annual international ACM SIGIR conference on Research and development in informaion retrieval*, pages 119–126. ACM, 2003.
- [94] T. Joachims. Optimizing search engines using clickthrough data. In *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 133–142. ACM, 2002.

- [95] M. Kalayeh, H. Idrees, and M. Shah. NMF-KNN: Image Annotation Using Weighted Multi-view Non-negative Matrix Factorization. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 184–191, June 2014.
- [96] A. Karpathy and L. Fei-Fei. Deep visual-semantic alignments for generating image descriptions. *arXiv preprint arXiv:1412.2306*, 2014.
- [97] A. Khosla, T. Zhou, T. Malisiewicz, A. A. Efros, and A. Torralba. Undoing the damage of dataset bias. In *European conference on computer vision (ECCV)*, pages 158–171, 2012.
- [98] R. Kiros, R. Salakhutdinov, and R. S. Zemel. Unifying visual-semantic embeddings with multimodal neural language models. *arXiv preprint arXiv:1411.2539*, 2014.
- [99] A. Krizhevsky, I. Sutskever, and G. Hinton. ImageNet classification with deep Convolutional Neural Networks. In *Annual conference on neural information processing systems (NIPS)*, 2012.
- [100] B. Kulis, K. Saenko, and T. Darrell. What you saw is not what you get: Domain adaptation using asymmetric kernel transforms. In *IEEE conference on computer vision and pattern recognition (CVPR)*, 2011.
- [101] A. Kurakin, I. Goodfellow, and S. Bengio. Adversarial examples in the physical world. *arXiv preprint arXiv:1607.02533*, 2016.
- [102] C. Lampert, H. Nickisch, and S. Harmeling. Learning to detect unseen object classes by between-class attribute transfer. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 951–958, June 2009.
- [103] C. H. Lampert, H. Nickisch, and S. Harmeling. Attribute-based classification for zero-shot visual object categorization. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 36(3):453–465, 2014.
- [104] V. Lavrenko, R. Manmatha, and J. Jeon. A model for learning the semantics of pictures. In *Advances in neural information processing systems*, page None, 2003.

- [105] R. Lebrecht, P. Pinheiro, and R. Collobert. Phrase-based Image Captioning. In D. Blei and F. Bach, editors, *Proceedings of the 32nd International Conference on Machine Learning (ICML-15)*, pages 2085–2094. JMLR Workshop and Conference Proceedings, 2015.
- [106] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998. tex.publisher: IEEE.
- [107] S. Lee, W. De Neve, and Y. M. Ro. Visually weighted neighbor voting for image tag relevance learning. *Multimedia tools and applications*, 72(2):1363–1386, 2014.
- [108] X. Li, C. G. Snoek, and M. Worring. Learning social tag relevance by neighbor voting. *Multimedia, IEEE Transactions on*, 11(7):1310–1322, 2009.
- [109] X. Li, T. Uricchio, L. Ballan, M. Bertini, C. G. M. Snoek, and A. Del Bimbo. Socializing the Semantic Gap: A Comparative Survey on Image Tag Assignment, Refinement and Retrieval. *arXiv:1503.08248 [cs]*, Mar. 2015.
- [110] Z. Li and J. Chen. Superpixel segmentation using linear spectral clustering. In *IEEE conference on computer vision and pattern recognition (CVPR)*, pages 1356–1363, 2015.
- [111] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollr, and C. L. Zitnick. Microsoft coco: Common objects in context. In *European conference on computer vision (ECCV)*, pages 740–755. Springer, 2014.
- [112] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollr, and C. L. Zitnick. Microsoft coco: Common objects in context. In *European conference on computer vision*, pages 740–755, 2014. tex.organization: Springer.
- [113] Y. Liu, X. Chen, C. Liu, and D. Song. Delving into transferable adversarial examples and black-box attacks. In *Proceedings of 5th international conference on learning representations*, 2017.
- [114] J. Long, E. Shelhamer, and T. Darrell. Fully convolutional networks for semantic segmentation. In *IEEE conference on computer vision and pattern recognition (CVPR)*, pages 3431–3440, 2015.
- [115] M. Long, Y. Cao, J. Wang, and M. I. Jordan. Learning transferable features with deep adaptation networks. In *International conference on machine learning (ICML)*, 2015.

- [116] D. G. Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision (IJCV)*, 60(2):91–110, 2004. tex.publisher: Springer.
- [117] J. Lu, H. Sibai, and E. Fabry. Adversarial examples that fool detectors. *arXiv preprint arXiv:1712.02494*, 2017.
- [118] J. Lu, H. Sibai, E. Fabry, and D. Forsyth. No need to worry about adversarial examples in object detection in autonomous vehicles. *arXiv preprint arXiv:1707.03501*, 2017.
- [119] J. Lu, H. Sibai, E. Fabry, and D. Forsyth. Standard detectors aren’t (currently) fooled by physical adversarial stop signs. *arXiv preprint arXiv:1710.03337*, 2017.
- [120] A. Makadia, V. Pavlovic, and S. Kumar. Baselines for image annotation. *International Journal of Computer Vision*, 90(1):88–105, 2010.
- [121] T. Mei, Y. Wang, X.-S. Hua, S. Gong, and S. Li. Coherent image annotation by learning semantic distance. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–8. IEEE, 2008.
- [122] T. Mikolov, K. Chen, G. Corrado, and J. Dean. Efficient Estimation of Word Representations in Vector Space. *arXiv:1301.3781 [cs]*, Jan. 2013.
- [123] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119, 2013.
- [124] T. Mikolov, W.-t. Yih, and G. Zweig. Linguistic Regularities in Continuous Space Word Representations. In *HLT-NAACL*, pages 746–751, 2013.
- [125] F. Monay and D. Gatica-Perez. PLSA-based image auto-annotation: constraining the latent space. In *Proceedings of the 12th annual ACM international conference on Multimedia*, pages 348–351. ACM, 2004.
- [126] S.-M. Moosavi-Dezfooli, A. Fawzi, O. Fawzi, and P. Frossard. Universal adversarial perturbations. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1765–1773, 2017.

- [127] R. Mottaghi, X. Chen, X. Liu, N.-G. Cho, S.-W. Lee, S. Fidler, R. Urtasun, and A. Yuille. The role of context for object detection and semantic segmentation in the wild. In *IEEE conference on computer vision and pattern recognition (CVPR)*, pages 891–898, 2014.
- [128] E. Moxley, T. Mei, and B. S. Manjunath. Video annotation through search and graph reinforcement mining. *Multimedia, IEEE Transactions on*, 12(3):184–193, 2010.
- [129] Z. Murez, S. Kolouri, D. Kriegman, R. Ramamoorthi, and K. Kim. Image to image translation for domain adaptation. *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.
- [130] M. Nair, M. Gillroy, N. Jose, and J. Davies. i-Surveillance crime monitoring and prevention using neural networks. *International Research Journal of Engineering and Technology*, 5, Mar. 2018.
- [131] G. Neuhold, T. Ollmann, S. R. Buló, and P. Kotschieder. The mapillary vistas dataset for semantic understanding of street scenes. In *IEEE international conference on computer vision (ICCV)*, pages 22–29, 2017.
- [132] T. Nguyen-Phuoc, C. Li, S. Balaban, and Y. Yang. RenderNet: A deep convolutional network for differentiable rendering from 3D shapes. *arXiv preprint arXiv:1806.06575*, 2018.
- [133] Z. Niu, G. Hua, X. Gao, and Q. Tian. Semi-supervised relational topic model for weakly annotated image recognition in social media. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4233–4240. IEEE, 2014.
- [134] H. Noh, S. Hong, and B. Han. Learning deconvolution network for semantic segmentation. In *IEEE international conference on computer vision (ICCV)*, pages 1520–1528, 2015.
- [135] M. Norouzi, T. Mikolov, S. Bengio, Y. Singer, J. Shlens, A. Frome, G. S. Corrado, and J. Dean. Zero-shot learning by convex combination of semantic embeddings. *arXiv preprint arXiv:1312.5650*, 2013.
- [136] M. Palatucci, D. Pomerleau, G. E. Hinton, and T. M. Mitchell. Zero-shot learning with semantic output codes. In *Advances in neural information processing systems*, pages 1410–1418, 2009.
- [137] S. J. Pan, J. T. Tsang, Ivor W. and Kwok, and Q. Yang. Domain adaptation via transfer component analysis. *Transactions on Neural Networks*, 22(2):199 – 210, 2011. tex.publisher: IEEE.

- [138] S. J. Pan and Q. Yang. A survey on transfer learning. *Transactions on Knowledge and Data Engineering*, 22(10):1345–1359, 2010. tex.publisher: IEEE.
- [139] X. Pan, P. Luo, J. Shi, and X. Tang. Two at once: Enhancing learning and generalization capacities via IBN-Net. In *European conference on computer vision (ECCV)*, pages 464–479, 2018.
- [140] G. Papandreou, L.-C. Chen, K. P. Murphy, and A. L. Yuille. Weakly- and semi-supervised learning of a deep convolutional network for semantic image segmentation. In *IEEE international conference on computer vision (ICCV)*, pages 1742–1750, 2015.
- [141] N. Papernot, P. McDaniel, I. Goodfellow, S. Jha, Z. B. Celik, and A. Swami. Practical black-box attacks against machine learning. In *Proceedings of the 2017 ACM on asia conference on computer and communications security*, pages 506–519, 2017. tex.organization: ACM.
- [142] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala. Pytorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. dAlché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, pages 8024–8035. Curran Associates, Inc., 2019.
- [143] V. M. Patel, R. Gopalan, R. Li, and R. Chellappa. Visual domain adaptation: A survey of recent advances. *Signal Processing Magazine*, 32(3):53–69, 2015. tex.publisher: IEEE.
- [144] D. Pathak, P. Krahenbuhl, and T. Darrell. Constrained convolutional neural networks for weakly supervised segmentation. In *IEEE international conference on computer vision (ICCV)*, pages 1796–1804, 2015.
- [145] C. Pellerin. Project maven to deploy computer algorithms to war zone by year’s end. July 2017.
- [146] X. Peng and K. Saenko. Synthetic to real adaptation with deep generative correlation alignment networks. *arXiv preprint arXiv:1701.05524*, 2017. tex.date-added: 2017-03-14 21:42:36 +0000 tex.date-modified: 2017-03-14 21:42:36 +0000.

- [147] X. Peng, B. Usman, N. Kaushik, J. Hoffman, D. Wang, and K. Saenko. Visda: The visual domain adaptation challenge. *arXiv preprint arXiv:1710.06924*, 2017.
- [148] J. Pennington, R. Socher, and C. D. Manning. Glove: Global vectors for word representation. *Proceedings of the Empirical Methods in Natural Language Processing (EMNLP 2014)*, 12, 2014.
- [149] F. Perronnin and C. Dance. Fisher kernels on visual vocabularies for image categorization. In *IEEE conference on computer vision and pattern recognition (CVPR)*, pages 1–8, 2007. tex.organization: IEEE.
- [150] F. Perronnin, J. Snchez, and T. Mensink. Improving the fisher kernel for large-scale image classification. In *European conference on computer vision (ECCV)*, pages 143–156, 2010.
- [151] P. O. Pinheiro and R. Collobert. From image-level to pixel-level labeling with convolutional networks. In *IEEE conference on computer vision and pattern recognition (CVPR)*, pages 1713–1721, 2015.
- [152] J. Redmon and A. Farhadi. YOLO9000: Better, faster, stronger. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 6517–6525, 2017. tex.organization: IEEE.
- [153] J. Redmon and A. Farhadi. YOLOv3: An incremental improvement. *arXiv preprint arXiv:1804.02767*, 2018.
- [154] S. Ren, K. He, R. Girshick, and J. Sun. Faster R-CNN: Towards real-time object detection with region proposal networks. In *Advances in neural information processing systems*, pages 91–99, 2015.
- [155] S. R. Richter, Z. Hayder, and V. Koltun. Playing for benchmarks. In *International conference on computer vision*, volume 2, 2017.
- [156] S. R. Richter, V. Vineet, S. Roth, and V. Koltun. Playing for data: Ground truth from computer games. In *European conference on computer vision (ECCV)*, pages 102–118. Springer, 2016.
- [157] R. Romijnders, P. Meletis, and G. Dubbelman. A domain agnostic normalization layer for unsupervised adversarial domain adaptation. *IEEE Winter Conference on Applications of Computer Vision*, 2019.

- [158] G. Ros, L. Sellart, J. Materzynska, D. Vazquez, and A. M. Lopez. The synthia dataset: A large collection of synthetic images for semantic segmentation of urban scenes. In *IEEE conference on computer vision and pattern recognition (CVPR)*, pages 3234–3243, 2016.
- [159] G. Ros, L. Sellart, J. Materzynska, D. Vazquez, and A. M. Lopez. The synthia dataset: A large collection of synthetic images for semantic segmentation of urban scenes. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3234–3243, 2016.
- [160] G. Ros, S. Stent, P. F. Alcantarilla, and T. Watanabe. Training constrained deconvolutional networks for road scene semantic segmentation. *arXiv preprint arXiv:1604.01545*, 2016.
- [161] A. Ruano. DeepGTAV. *GitHub repository*, 2017. tex.publisher: GitHub.
- [162] D. E. Rumelhart, G. E. Hinton, and R. J. Williams. Learning internal representations by error propagation. Technical report, DTIC Document, 1985.
- [163] K. Saenko, B. Kulis, M. Fritz, and T. Darrell. Adapting visual category models to new domains. In *European conference on computer vision (ECCV)*, pages 213–226, 2010. tex.organization: Springer.
- [164] K. Saito, Y. Ushiku, T. Harada, and K. Saenko. Adversarial dropout regularization. *International Conference on Learning representations (ICLR)*, 2018.
- [165] K. Saito, K. Watanabe, Y. Ushiku, and T. Harada. Maximum classifier discrepancy for unsupervised domain adaptation. *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.
- [166] F. S. Saleh, M. S. Aliakbarian, M. Salzmann, L. Petersson, and J. M. Alvarez. Effective use of synthetic data for urban scene semantic segmentation. In *European conference on computer vision (ECCV)*, pages 86–103, 2018. tex.organization: Springer, Cham.
- [167] S. Sankaranarayanan, Y. Balaji, A. Jain, S. N. Lim, and R. Chellappa. Unsupervised domain adaptation for semantic segmentation with GANs. *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.
- [168] R. R. Selvaraju, M. Cogswell, A. Das, R. Vedantam, D. Parikh, D. Batra, and others. Grad-cam: Visual explanations from deep networks via gradient-based localization. In *International conference on computer vision*, pages 618–626, 2017.

- [169] A. Shafaei, J. J. Little, and M. Schmidt. Play and learn: using video Games to train computer vision models. *arXiv preprint arXiv:1608.01745*, 2016.
- [170] S. Shah, D. Dey, C. Lovett, and A. Kapoor. AirSim: High-fidelity visual and physical simulation for autonomous vehicles. In *Field and service robotics*, 2017. tex.eprint: arXiv:1705.05065.
- [171] J. Shotton, M. Johnson, and R. Cipolla. Semantic texton forests for image categorization and segmentation. In *2008 IEEE conference on computer vision and pattern recognition*, pages 1–8, June 2008. ISSN: 1063-6919.
- [172] A. Shrivastava, T. Pfister, O. Tuzel, J. Susskind, W. Wang, and R. Webb. Learning from simulated and unsupervised images through adversarial training. *arXiv preprint arXiv:1612.07828*, 2016. tex.date-added: 2017-03-14 21:45:02 +0000 tex.date-modified: 2017-03-14 21:45:02 +0000.
- [173] B. Sigurbjrnsson and R. Van Zwol. Flickr tag recommendation based on collective knowledge. In *Proceedings of the 17th international conference on World Wide Web*, pages 327–336. ACM, 2008.
- [174] K. Simonyan, A. Vedaldi, and A. Zisserman. Deep inside convolutional networks: Visualising image classification models and saliency maps. *arXiv preprint arXiv:1312.6034*, 2013.
- [175] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.
- [176] R. Socher, J. Bauer, C. D. Manning, and A. Y. Ng. Parsing with compositional vector grammars. In *In Proceedings of the ACL conference*. Citeseer, 2013.
- [177] R. Socher, M. Ganjoo, C. D. Manning, and A. Ng. Zero-shot learning through cross-modal transfer. In *Advances in neural information processing systems*, pages 935–943, 2013.
- [178] B. Sun, J. Feng, and K. Saenko. Return of frustratingly easy domain adaptation. In *AAAI conference on artificial intelligence (AAAI)*, 2016.
- [179] B. Sun and K. Saenko. From virtual to reality: Fast adaptation of virtual object detectors to real domains. In *BMVA british machine vision conference (BMVC)*, 2014. tex.date-modified: 2017-03-13 20:43:06 +0000.

- [180] C. Szegedy, S. Ioffe, V. Vanhoucke, and A. Alemi. Inception-v4, inception-resnet and the impact of residual connections on learning. *arXiv preprint arXiv:1602.07261*, 2016.
- [181] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, and R. Fergus. Intriguing properties of neural networks. *arXiv preprint arXiv:1312.6199*, 2013.
- [182] A. Tariq and H. Foroosh. Feature-Independent Context Estimation for Automatic Image Annotation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1958–1965, 2015.
- [183] S. Tellex, B. Katz, J. Lin, A. Fernandes, and G. Marton. Quantitative evaluation of passage retrieval algorithms for question answering. In *Proceedings of the 26th annual international ACM SIGIR conference on Research and development in informaion retrieval*, pages 41–47. ACM, 2003.
- [184] J. Tighe and S. Lazebnik. Superparsing: scalable nonparametric image parsing with superpixels. In *European conference on computer vision (ECCV)*, pages 352–365, 2010. tex.organization: Springer.
- [185] T. Tommasi, N. Patricia, B. Caputo, and T. Tuytelaars. A deeper look at dataset bias. *CoRR*, arXiv:1505.01257, 2015.
- [186] A. Torralba and A. A. Efros. Unbiased look at dataset bias. In *IEEE conference on computer vision and pattern recognition (CVPR)*, 2011.
- [187] J. Tremblay, A. Prakash, D. Acuna, M. Brophy, V. Jampani, C. Anil, T. To, E. Cameracci, S. Bochoon, and S. Birchfield. Training deep networks with synthetic data: Bridging the reality gap by domain randomization. In *Proceedings of the IEEE conference on computer vision and pattern recognition workshops*, pages 969–977, 2018.
- [188] Y.-H. Tsai, W.-C. Hung, S. Schuler, K. Sohn, M.-H. Yang, and M. Chandraker. Learning to adapt structured output space for semantic segmentation. *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.
- [189] G. Tsoumakas and I. Katakis. Multi-label classification: An overview. *Dept. of Informatics, Aristotle University of Thessaloniki, Greece*, 2006.

- [190] E. Tzeng, J. Hoffman, T. Darrell, and K. Saenko. Simultaneous deep transfer across domains and tasks. In *IEEE international conference on computer vision (ICCV)*, pages 4068–4076, 2015.
- [191] E. Tzeng, J. Hoffman, N. Zhang, K. Saenko, and T. Darrell. Deep domain confusion: Maximizing for domain invariance. *CoRR*, arXiv:1412.3474, 2014.
- [192] L. Van der Maaten and G. Hinton. Visualizing data using t-SNE. *Journal of Machine Learning Research*, 9(2579-2605):85, 2008.
- [193] G. Varol, J. Romero, X. Martin, N. Mahmood, M. J. Black, I. Laptev, and C. Schmid. Learning from synthetic humans. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4627–4635. IEEE, 2017.
- [194] D. Vazquez, A. M. Lopez, J. Marn, D. Ponsa, and D. Gernimo. Virtual and real world adaptation for pedestrian detection. *Transactions of Pattern Recognition and Machine Analyses (PAMI)*, 36(4):797–809, 2014. tex.date-modified: 2017-03-13 20:42:49 +0000 tex.publisher: IEEE.
- [195] K. Weiss, T. M. Khoshgoftaar, and D. Wang. A survey of transfer learning. *Journal of Big Data*, 3(1):9, 2016. tex.publisher: Nature Publishing Group.
- [196] J. Weston, S. Bengio, and N. Usunier. Large scale image annotation: learning to rank with joint word-image embeddings. *Machine learning*, 81(1):21–35, 2010.
- [197] J. Weston, S. Bengio, and N. Usunier. Wsabie: Scaling up to large vocabulary image annotation. In *IJCAI*, volume 11, pages 2764–2770, 2011.
- [198] Z. Wu, X. Han, Y.-L. Lin, M. G. Uzunbas, T. Goldstein, S. N. Lim, and L. S. Davis. DCAN: Dual channel-wise alignment networks for unsupervised scene adaptation. *European Conference on Computer Vision (ECCV)*, 2018.
- [199] Z. Wu, C. Shen, and A. v. d. Hengel. Wider or Deeper: Revisiting the ResNet Model for Visual Recognition. *arXiv preprint arXiv:1611.10080*, 2016.
- [200] F. Xia, T.-Y. Liu, J. Wang, W. Zhang, and H. Li. Listwise approach to learning to rank: theory and algorithm. In *Proceedings of the 25th international conference on Machine learning*, pages 1192–1199. ACM, 2008.

- [201] C. Xie, J. Wang, Z. Zhang, Y. Zhou, L. Xie, and A. Yuille. Adversarial examples for semantic segmentation and object detection. In *International conference on computer vision*, 2017. tex.organization: IEEE.
- [202] H. Xu, Y. Gao, F. Yu, and T. Darrell. End-to-end learning of driving models from large-scale video datasets. *arXiv preprint arXiv:1612.01079*, 2016.
- [203] J. Xu, S. Ramos, D. Vazquez, and A. Lpez. Hierarchical adaptive structural SVM for domain adaptation. *CoRR*, arXiv:1408.5400, 2014. tex.date-modified: 2017-03-13 20:48:04 +0000.
- [204] O. Yakhnenko and V. Honavar. Annotating images and image objects using a hierarchical dirichlet process model. In *Proceedings of the 9th International Workshop on Multimedia Data Mining: held in conjunction with the ACM SIGKDD 2008*, pages 1–7. ACM, 2008.
- [205] J. Yosinski, J. Clune, Y. Bengio, and H. Lipson. How transferable are features in deep neural networks? In *Advances in neural information processing systems*, pages 3320–3328, 2014.
- [206] F. Yu, V. Koltun, and T. Funkhouser. Dilated residual networks. In *IEEE conference on computer vision and pattern recognition (CVPR)*, 2017.
- [207] M. D. Zeiler. ADADELTA: an adaptive learning rate method. *arXiv preprint arXiv:1212.5701*, 2012.
- [208] C. Zhang, L. Wang, and R. Yang. Semantic segmentation of urban scenes using dense depth maps. In *European conference on computer vision (ECCV)*, pages 708–721, 2010. tex.organization: Springer.
- [209] F. Zhang, J. Leitner, M. Milford, and P. Corke. Sim-to-real transfer of visuo-motor policies for reaching in clutter: Domain randomization and adaptation with modular networks. *arXiv preprint arXiv:1709.05746*, 2017.
- [210] Y. Zhang, P. David, and B. Gong. Curriculum domain adaptation for semantic segmentation of urban scenes. In *IEEE international conference on computer vision (ICCV)*, volume 2, page 6, Oct. 2017. tex.number: 5.
- [211] Y. Zhang, Z. Qiu, T. Yao, D. Liu, and T. Mei. Fully convolutional adaptation networks for semantic segmentation. In *IEEE conference on computer vision and pattern recognition (CVPR)*, June 2018.

- [212] H. Zhao, J. Shi, X. Qi, X. Wang, and J. Jia. Pyramid Scene Parsing Network. *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.
- [213] J.-Y. Zhu, T. Park, P. Isola, and A. A. Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. In *IEEE conference on computer vision and pattern recognition (CVPR)*, pages 2223–2232, 2017.
- [214] X. Zhu, W. Nejdl, and M. Georgescu. An adaptive teleportation random walk model for learning social tag relevance. In *Proceedings of the 37th international ACM SIGIR conference on Research & development in information retrieval*, pages 223–232. ACM, 2014.
- [215] X. Zhu, H. Zhou, C. Yang, J. Shi, and D. Lin. Penalizing top performers: Conservative loss for semantic segmentation adaptation. *European Conference on Computer Vision (ECCV)*, 2, 2018.
- [216] W. Y. Zou, R. Socher, D. M. Cer, and C. D. Manning. Bilingual Word Embeddings for Phrase-Based Machine Translation. In *EMNLP*, pages 1393–1398, 2013.
- [217] Y. Zou, Z. Yu, B. Vijaya Kumar, and J. Wang. Unsupervised domain adaptation for semantic segmentation via class-balanced self-training. In *European conference on computer vision (ECCV)*, Sept. 2018.