August 31, 2019

This chapter is just a collection of notes on papers on **Syn2Real Domain Adaptation**.

1 Domain Adaptation for Structured Output via Discriminative Patch Representation

see [**?**]

1.1 Abstract

- labeling data is expensive
- therefore propose domain adaptation method to adapt labeled source data to unlabeled target domain (e.g. GTA5 (playing for data) to city-scapes)
- learn discriminative feature representations of patches based on label histograms in the source domain, through construction of clustered space
- then use adversarial learning sscheme to push feature representations in target patches to the closer distributions in source ones
- can integrate a global alignment process with this patch-level alignment and achieve state-of-the-art performance on semantic segmentation
- extensive ablation studies on numerous benchmark datasets with various settings (e.g. synth-to-real, cross-city)

1.2 Introduction

- pixel-level annotation of ground truth expensive. e.g. road-scene iamges of different cities may have various appearance distributions, differences over time and weather
- existing state-of-the-art methods use feature-level or output space adaptation, exploiut global distribution alignment, such as spatial layout, but these might differ significantly between two domains due to differences in camera poses or field of view

- authours instead match patches that are more likely to be shared across domains regardless of where they are located
- consider label histograms as a factor (Kulkarni et al., 2015; Odena et al., 2017) and learn discriminative representations for patches to relax high-variation problem among them
- use this to better align patches between source and target domains
- utilize two adversarial modules to align global/patch-level distributions
- global one based on output space adaptation (Tsai et al. 2018)
- take source domain labels and extract label histogram as a patch-level representation
- then apply K-means clustering to group extracted patch representations into K clusters **TODO: read this part again for better understanding (page 2)**

1.3 domain adaptation for structured output

- given source and target images $I_s, I_t \in \mathbb{R}^{H\times W\times 3}$ and source labels Y_s, the goal is to align predicted output distribution O_t of target data with source distribution O_s
- use loss function for supervised learning on source data to predict the structured output, adversarial loss is adopted to align the global distribution
- further incorporate classification loss in a clustered space to learn patch-level discriminative representations F_s from source output distribution O_s. For target data another adversarial loss is used to align patch-level distributions between F_s and F_t, where the goal is to push F_t to be closer to distributon of F_s.
- objective function :

$$\mathcal{L}_{\text{total}}(I_s, I_t, Y_s, \Gamma(Y_s)) = \mathcal{L}_s + \lambda_d \mathcal{L}_d + \lambda^g_{\text{adv}}\mathcal{L}^g_{\text{adv}} + \lambda^l_{\text{adv}}\mathcal{L}^l_{\text{adv}} \qquad (1)$$

  where $\mathcal{L}_s$ and $\mathcal{L}_d$ are supervised loss function for learning structured prediction and discriminative representation on source data. Γ denotes clustering process on ground truth label distribution. $\mathcal{L}^g_{\text{adv}}, \mathcal{L}^l_{\text{adv}}$ denote global and patch-level adversarial loss. λ's are weights for the different loss function
- $\mathcal{L}_s$ can be optimized by fully-convolutional network $\mathbf{G}$ that predicts the structured output with the loss summed over the spatial map indexed with h, w and number of categories C:

$$\mathcal{L}_s(I_s, Y_s; \mathbf{G}) = -\sum_{h,w}\sum_{c\in C} Y_s^{(h,w,c)} \log(O_s^{(h,w,c)}) \qquad (2)$$

where $O_s = \mathbf{G}(I_s) \in (0,1)$ is the predicted output distribution through softmax function and is up-sampled to the size of the input image.

- with discriminator $\mathbf{D}_g$:

$$\mathcal{L}^g_{\text{adv}}(I_s, I_t; \mathbf{G}, \mathbf{D}_g) = \sum_{h,w} \mathbb{E}[\log \mathbf{D}_g(O_s)^{(h,w,1)}] + \mathbb{E}[\log(1 - \mathbf{D}_g(O_t)^{(h,w,1)})] \tag{3}$$

- optimize following min-max problem with inputs dropped for simplicity:

$$\min_{\mathbf{G}} \max_{\mathbf{D}_g} \mathcal{L}_s(\mathbf{G}) + \lambda^g_{\text{adv}} \mathcal{L}^g_{\text{adv}}(\mathbf{G}, \mathbf{D}_g) \tag{4}$$

- label histograms for patches: first randomly sample patches from source images, using a 2-by-2 grid on patches to extract spatial label histograms, and concatenate them into a vector, each histogram is a $2 \cdot 2 \cdot C$ dimensional vector. Second apply K-means clustering on these histograms, whereby the label for any patch can be assigned as the cluster center with the closest distance on the histogram
- add classification module $\mathbf{H}$ after the predicted output O_s, to simulate the procedure of constructin the label histogram and learn a discriminative representation
  learned representation: $F_s = \mathbf{H}(\mathbf{G}(I_s)) \in (0,1)^{U \times V \times K}$ (softmax function, K is number of clusters)
- learning process to construct clustered space formulated as cross-entropy loss:

$$\mathcal{L}_d(I_s, \Gamma(Y_s); \mathbf{G}, \mathbf{H}) = -\sum_{u,v} \sum_{k \in K} \Gamma(Y_s)^{(u,v,k)} \log(F_s^{(u,v,k)}) \tag{5}$$

- goal is now to align patches regardless of where they are located in the image (without spatial and neighborhood support)
- reshape F by concatenating the K-dimensional vectors along the spatial map, results in $U \cdot V$ independent data points
- this reshaped data is denoted as $\hat{F}$, adversarial objective:

$$\mathcal{L}^l_{\text{adv}}(I_s, I_t; \mathbf{G}, \mathbf{H}, \mathbf{D}_l) = \sum_{u,v} \mathbb{E}[\log \mathbf{D}_l(\hat{F}_s)^{(u,v,1)}] + \mathbb{E}[\log(1 - \mathbf{D}_l(\hat{F}_t)^{(u,v,1)})] \tag{6}$$

  where $\mathbf{D}_l$ is the discriminator to classify whether the feature representation $\hat{F}$ is from source or target domain
- integrate (3.5) and (3.6) into min-max problem in 3.4:

$$\min_{\mathbf{G},\mathbf{H}} \max_{\mathbf{D}_g,\mathbf{D}_l} \mathcal{L}_s(\mathbf{G}) + \lambda_d \mathcal{L}_d(\mathbf{G}, \mathbf{H}) + \lambda^g_{\text{adv}}, \mathcal{L}^g_{\text{adv}}(\mathbf{G}, \mathbf{D}_g) + \lambda^l_{\text{adv}} \mathcal{L}^l_{\text{adv}}(\mathbf{G}, \mathbf{H}, \mathbf{D}_l) \tag{7}$$

2 Effective Use of Synthetic Data for Urban Scene Semantic Segmentation

see [**?**]

- foreground and background classes are not affected the same way by domain shifts
- foreground classes should be treated in a detection based manner as their shape looks natural even though their texture in synthetic images is not photo-realistic
- drawback of deep neural networks is need for massive amount of training data
- training on synthetic only makes models perform bad in the real world
- domain adaptation methods improve this but still require large sets of real images
- model cannot be trained off-line on synthetic data and work well when deployed into a new, real-world environment
- observation: not all classes suffer from the same type and degree of perceptual differences
- background texture looks more realistic than foreground, nevertheless foreground object shape look natural
- therefore should be treated differently
- use semantic segmentation on background classes because of texture realism
- use object detectors for foreground classes
- main discrimination between fore and background objects is shape
- trained seperately a DeepLab and Mask R-CNN **TODO: cite** for object detection, followed by binary segmentation and class prediction, on synthetic data
- compare mIoU on foreground classes of Cityscapes
- outperforms semantic segmentation model on every class except *motorcycle*
- from this observation: model that combines foreground masks produced by Mask R-CNN with pixel-wise predictions of DeepLab semantic segmentation network
- outperforms state-of-the-art domain adaptation techniques and can be further improved by making use of unsupervised real images
- **VEIS** (Virtual Environment for Instance Segmentation) proposed aswell, based on Unity3D

- automatically annotates synthetic images with instance-level segmentation for foreground classes
- when used with the proposed detector-based approach this data allows to boost semantic segmentation performance

2.1 related work

- domain adaptation generally aims to reduce the gap between the feature distributions of the two domains (synthetic, real)

2.2 Method

- use VGG16-based DeepLab model for background classes (with large field of view and dilated convolution layers)
- train on GTA5 dataset (background classes look photo-realistic)
- cross-entropy loss between network's predictions and ground-truth pixel-wise annotations of the sythetic images
- trained on fore- and background classes but foreground predictions are mostly discarded by the proposed approach
- use standard Mask R-CNN (object detection, binary mask extraction together with object classification) **TODO: look at citation**
- fuse fore- and background predictions
- sort predicted segments according to confidence score, if current segment candidate overlaps with previous segment, pixels are removed in the overlapping region
- this yields a semantic segmentation map that only contains foreground classes and has large number of holes where no foreground object was found. Every pixel that is not already assigned to foreground class takes the label with highest probability at that pixel location in the DeepLab result
- remember: NO real data used during training of this method
- to extend approach for unsupervised training on real images: tread predictions of their method as pseudo ground truth for the real images. assign pixels that ware predicted as foreground classes by DeepLab model to an *ignore* label, so that they are not used for training.

2.3 Implementation Details

- DeepLab: SGD, learning rate starting at 25×10^{-5} with decrease factor of 10 every 40k iterations, momentum of 0.9, weight decy of 0.0005, mini-batches of size 1. weights initialized with those of VGG-16 classifier, pre-trained

on ImageNet, due to limited GPU memory high resolution images were downsampled by a factor of 2 for training

- Mask R-CNN: implementation provided by Detectron framework **TODO: cite**, train an end-to-end Mask R-CNN model with 64× 4d ResNeXt-101-FPN backbone, pre-trained on ImageNet, on the synthetic VEIS dataset. mini-batch of size 1, train model for 200k iterations, starting learning rate of 0.001, reducing to 0.0001 after 100k iterations

3 Exemplar Guided Unsupervised Image-to-Image Translation with Semantic Consistency

see [**?**]

- EGSC-IT network conditions the translation process on an exemplar image in the target domain
- assumption: image comprises of a content component shared across domains, and style component specific to each domain
- under guidance of exemplar from target domain apply Adaptive Instance Normalization to the shared content component, which allows to transfer the style information of the target domain to the source domain
- introduce feature masks that provide coarse semantic guidance without requiring the use of any semantic labels to avoid semantic inconsistencies during translation
- image-to-image (I2I) translation: task of mapping an image from a source domain to a target domain (e.g. semantic maps to real images, grey-scale to color, low-resolution to high-res)
- other approaches assume one-to-one mapping (e.g. cycleGAN) and fail to capture multimodal nature of image distribution within the target domain (many-to-many mapping, e.g. diffferent color and style of shoes in sketch-to-image translation, different seasons in syn2real street view translation)
- assume an image is composed of two disentangled representations: domain-shared representation that models the content in the image, second domain-specific representation that contains the style information
- unclear which style (time-of-day/season) to pick during image translation process, Isola et al. 2017 and Zhu et al. 2017b **TODO: cite** show that using random noise for this can lead to mode collapse issues
- instead the image translation process is conditioned on arbitrary image in the target domain (i.e. an exemplar)

- like this EGSC-IT is enabled for multimodal (i.e. many-to-many) image translations, but also allows for explicit control over translation process depending on the exemplars used as guidance
- use weight sharing architecture proposed in UNIT (Liu et al. 2017 **TODO: cite**), but instead of single latent space shared by both domains, a two component one is used (domain-shared component that contains semantic information (objects' category, shape, spacial layout), domain-specific style component that contains style information (texture, color)
- adaptive instance normalization (AdaIN) (Huang & Belongie 2017 **TODO: cite**) is applied to the shared content component of the source domain image using AdaIN parameters computed from the target domaoin exemplar
- directly applying AdaIN to the feature maps of the shared content component would mix up all objects and scenes in the image, making the image translation prone to failure when a n image contains diverse objects and scenes, therefore semantic labels as an additional form of supervision is used **TODO: cite works that do this**.
- instead of using data with labor-intensive annotations use computed feature masks
- feature masks: approximately decoupling different semantic categories in an unsupervised way under guidance of perceptual loss and adversarial loss

3.1 Method

- weight sharing: to map image pairs (one from source domain, other target) to shared latent space, the last layer in the encoders (E_A, E_B) of the VAE-GAN and the first layer in the generators (G_A, G_B) share their weights. (see **TODO: cite UNIT, Liu et al. 2017**)
- **TODO: exemplar-based AdaIN for domain-specific style**
- *Network architecture*: 1) two encoders E_A, E_B, each consisting of serveral strided convolutional layers and serveral residual blocks to compute the shared content component, 2) feature mask network and an AdaIN network, F_A, F_B for $A \to B$ translation ($B \to A$ vise versa)have same architecture as the Encoder above **TODO: above?** except for weight-sharing layers. 3) Two Generators, G_A, G_B, almost symmetric to the Encoders except that the up-sampling is done by transposed convolutional layers. 4) Two Discriminators, D_A, D_B fully-convolutional networks containing a stack of convolutional layers. 5) A VGG sub-network (Simonyan & Zisserman, 2015 **TODO: cite**), VGG, that contains the first few layers (up to relu5_1) of pre-trained VGG-19 (**TODO: cite Simonyan & Zisserman, 2015**), which is used to calculate perceptual losses.
- Although UNIT is used as baseline framework, it can be replaced with any baseline framework with similar functionality

3.2 Learning

- learning procedure of EGSC-IT contains VAEs, GANs, cycle-consistecy and perceptual losses.
- for more stable training, the feature mask network and AdaIN network gets pre-trained for each domain seperately within a VAE-GAN architecture, and use encoder part as fixed feature extractors (F_A, F_B) for the remaining training
- overall loss:

$$\mathcal{L}(E_A, G_A, D_A, E_B, G_B, D_B) = \mathcal{L}_{\text{VAE}_\text{A}}(E_A, G_A) + \mathcal{L}_{\text{GAN}_\text{A}}(E_A, G_A, D_A) + \mathcal{L}_{\text{CC}_\text{A}}(E_A, G_A, E_B, G_B) \tag{8}$$

$$+ \mathcal{L}_{\text{VAE}_\text{B}}(E_B, G_B) + \mathcal{L}_{\text{GAN}_\text{B}}(E_B, G_B, D_B) + \mathcal{L}_{\text{CC}_\text{B}}(E_A, G_A, E_B, G_B) \tag{9}$$

$$+ \mathcal{L}_{\text{P}}(E_A, G_A, E_B, G_B) \tag{10}$$

- VAEs, GANs and cycle-consistency losses identical to the ones used in **TODO: cite Liu et al., 2017**
- **TODO: add rest**

3.3 Experiments

- translated between GTA5 and Berkeley Deep Drive (BDD) (**TODO: cite both**)
- Similar to FCN-score used by **TODO: cite isola et al 2017** use the semantic segmentation performance to quantitatively evaluate the image translation quality
- first translate images from GTA5 dataset to arbitrary image in BDD
- only generate images of size 256×512 due to GPU memory limitations
- then train single-scale Deeplab model (**TODO: cite Chen et al. 2018**) on the translated images and test it on BDD test set
- get mIoU scores

3.4 Discussion

since method doesn't use any semantic segmentation labels nor paired data, there are some artifacts in the results for some hard cases.
e.g. night $\rightarrow$ day translation more challenging than day $\rightarrow$ day, therefore sometimes hard for the model to understand semantics in such cases.
In the future it would be interesting to extend the method to semi-supervised setting to benefit from presence of some fully-labeled data

4 Exploiting Semantics in Adversarial Training for Image-Level Domain Adaptation

see [**?**]

- ResNet generator, U-Net discriminator
- 300k training iterations
- Adam optimizer, 0.0001 learning rate, batch size 2
- images cropped to 512x512
- images from GTA+annotations thereof, only images of cityscapes
- uses structure of CycleGAN
- uses discriminators also as semantic segmentation networks
- features extracted by the last encoder layer used to generate both semantic label map and domain classification score

Training

- Standard Adversarial loss
- Semantic Discriminator Loss:

$$\mathcal{L}_{\text{sem}} = H(D_{S_{\text{sem}}}(G_{S\to T}(X_S)), Y_S) + H(D_{S_{\text{sem}}}(X_S), Y_S) \tag{11}$$

  with pixel-wise cross entropy loss $H(p, q)$

- Weighted Reconstruction Loss:

$$\mathcal{L}_{\text{rec}} = \|G_{S\to T}(G_{T\to S}(x_T)) - x_T\|_1 \tag{12}$$
$$+ (1-w)\|G_{T\to S}(G_{S\to T}(x_S)) - x_S\|_1 \tag{13}$$

  each pixel in source given sample weighted proportionally to probability of not belonging to its semantic class. weighting term acts as a regularization where the network usually fail adaptation introducing artifacts, forcing the least frequent classes to be reconstructed preserving input appearance. w is a weight mask with same resolution as source image. C is set of possible classes, each weight $w_{i,j}$ represents likelihood of a class among whole synthetic dataset:

$$w_{i,j} = \frac{n_{\text{pixel}} \in c}{n_{\text{pixel}}}, \quad c \in C \tag{14}$$

- generators and discriminator trained to minimize:

$$\mathcal{L}_D = -\mathcal{L}_{\text{adv}} + \lambda_{\text{sem}}\mathcal{L}_{\text{sem}} \tag{15}$$
$$\mathcal{L}_G = \mathcal{L}_{\text{adv}} + \lambda_{\text{sem}}\mathcal{L}_{\text{sem}} + \lambda_{\text{rec}}\mathcal{L}_{\text{rec}} \tag{16}$$

λ_{sem} and λ_{rec} are hyper-parameters that control relative importance of domain classification, weighted reconstruction and semantic segmentation. Use $\lambda_{\text{sem}} = 1$ and $\lambda_{\text{rec}} = 3$ across all experiments

5 FCNs in the Wild: Pixel-level Adversarial and Constraint-based Adaptation

see [?]

- semantic segmentation is critical visual recognition task for a variety of applications ranging from autonomous agent tasks, such as robotic navigation and self-driving cars, to mapping and categorizing the natural world
- challenges when adapting between visual domains for classification: changes in appearance, lighting, pose

5.1 work includes

- first unsupervised domain adaptation method for transferring semantic segmentation FCNs across image domains
- combination of global and local alignment methods, using global and category specific adaptation techniques
- align global statistics of source and target data using a convolutional domain adversarial training technique, using a novel extension of previous image-level classification approaches **TODO: look up citations**
- given a domain aligned representation space, introduce generalizable constrained multiple instance loss function, which expands on weak label learning, but can be applied to the target domain without any extra annotations and explicitly transfers category layout information from a labeled source dataset
- use GTA5 and SYNTHIA + CityScapes datasets for synth2real
- cross season adaptation within SYNTHIA
- adaptation across real world cities
- perform detailed quantitative analysis of cross-city adaptation within CityScapes
- contribute BDDS (Berkeley Deep Driving Segmentation), unconstrained drive-cam dataset for semantic segmentation
- show Cityscapes to BDDS
- show that adaptation algorithm improves target semantic segmentation performance without any target annotations

5.2 Related Work

- many recent approaches use fully convolutional networks (FCNs) for semantic segmentation, mapping input RGB space to semantic pixel space
- compelling because they allow direct end-to-end function that can be trained using back propagation
- original FCN formulation has been improved using dilated convolution (**TODO: look at citation**) and post-processing techniques, such as Markov/conditional random fields
- high cost of collecting pixel level supervision motivates weak labels (typically image-level tags defining presence / absence of each class)
- Multiple instance learning (MIL) **TODO: see citations**, reinforce confident predictions during learning process
- improves method suggested by **TODO: see citations** who use an EM algorithm to better model global properties of the images segments. Generalized by Pathak et al. who proposed a Constrained CNN, also able to model any linear constraints on the label space (i.e. presence / absence, percent cover)
- Hong et al. used auxiliary segmentation to generalize semantic segmentations to categories where only weak label information was available
- these methods all assume weak labels present during training time for both source and target domain
- authors consider strong supervision available in source domain, no supervision available in target domain

5.2.1 Domain Adaptation

- domain adaptation in computer vision focuses largely on image classification, with much work dedicated to generalizing across domain shift between stock photographs of objects and the same objects photographed in the world
- recent work include **TODO: some citations** which all learn a feature representation which encourages maximal confusion between the two domains
- other work **TODO: cite** aims to align features by minimizing the distance between their distribtuions in the two domains
- based on GAN Liu et al. proposed coupled GAN to learn a joint distribution of images from both source and target datasets
- other computer vision task detection: Hoffmann et al., domain adaptation system: explicitly modeling the representation shift between classification

and detection models, follow-up work which incorporated per-category adaptation using multiple instance learning. later converted into FCNs for evaluating semantic segmentation performance

5.3 Fully Convolutional Adaptation Models

- source domain S, images I_S, labels L_S, trained source only model for semantic segmentation which produces pixel-wise per-category score map $\phi_S(I_S)$
- goal to learn semantic segmentation model which is adapted for use on unlabeled target domain T, images I_T, no annotations, parameters of such a network $\phi_T(\cdot)$
- if no domain shift: just apply source model directly to target domain
- however: commonly a difference between distribution of source labeled domain and target test domain
- therefore: *unsupervised adaptation* approach
- first main shift: global changes may occur between two domains resulting in marginal distribution shift of corresponding feature space
- can occur in any domain, but most distinct in large domain shifts between very distinct domains (e.g. simulated and real domains)
- second main shift: due to category specific parameter changes. may result from individual categories having specific biases in the two domains (e.g. adapting between different cities: changes in appearance of signs and distribution of cars)
- propose unsupervised domain adaptation framework for adapting semantic segmentation models which directly tackles both the need for minimizing the global and the category specific shifts
- assumptions: source and target domain share same label space, source model achieves performance greater than chance on target domain
- introduce two new semantic segmentation loss objectives, one to minimize the global distribution distance, which operates over both source and target images, $\mathcal{L}_{\text{da}}(I_S, I_T)$, another to adapt category specific parameters using target images and transferring label statistics from source domain $P_{L_S}, \mathcal{L}_{\text{mi}}(I_T, P_{L_S})$
- to ensure to not diverge too far from source solution, which is known to be effective for final semantic segmentation task, continue to optimize the standard supervised segmentation objective on source domain $\mathcal{L}_{\text{seg}}(I_S, L_S)$

- goal: optimize joint objective:

$$\mathcal{L}(I_S, L_S, I_T) = \mathcal{L}_{\text{seg}}(I_S, L_S) \tag{17}$$
$$+ \mathcal{L}_{\text{da}}(I_S, I_T) + \mathcal{L}_{\text{mi}}(I_T, P_{L_S}) \tag{18}$$

- **TODO: add illustration Figure 2**
- source domain data to update standard supervised loss objective, trained using source pixel-wise annotations
- both source and target data are used without any category annotations within fully-convolutional domain adversarial training to minimize global distance of feature space between the two domains
- category specific updates using a constrained pixel-wise multiple instance learning objective is performed on the target images, with source category statistics used to determine the constraints
- use front-end dilated FCN **TODO: cite 33**, based on VGG16 **TODO: cite 31** as base model
- 16 conv layers, last three conv layer converted from fully connected layers, called fc_6, fc_7, fc_8, followed by 8 times bilinear up-sample layer to produce segmentation in the same resolution as input images

5.4 Global Domain Alignment

- recognition sought at pixel level, alignment of full image representations will marginalize out too much distribution information, limiting the alignment capability of the adversarial learning approach
- instead consider region corresponding to natural receptive field of each spatial unit in final representation layer (e.g. fc_7), as individual instances
- in doing so, the adversarial training procedure is supplied with the same information which is used to do final pixel prediction
- provides a more meaningful view of overall source and target pixel-space representation distribution distance which needs to be minimized
- let $\phi_{l-1}(\theta, I)$ denote output of last layer before pixel prediction according to network parameters θ
- $\mathcal{L}_{\text{da}}(I_S, I_T)$ consists of alternating minimization objectives
- one concerning parameters θ of representation space, under which source and target distance $\min d(\phi_{l-1}(\theta, I_S), \phi_{l-1}(\theta, I_T))$ will be minimized for given distance function $d(\cdot)$.
- second: estimating distance function through training domain classifier to distinguish instances of source and target domains

- let θ_D domain classifier parameters
- learn domain classifier to recognize difference between source and target regions and use that classifier to guide distance minimization of source and target representations
- let $\sigma(\cdot)$ denote softmax function, domain classifier predictions $p_{\theta_D}(x) = (\sigma(\phi(\theta_D, x)))$
- assuming output of layer $l-1$ has $H \times W$ spatial units, we can define domain classifier loss $\mathcal{L}_D$ as:

$$\mathcal{L}_D = -\sum_{I_S \in S}\sum_{h \in H}\sum_{w \in W} \log(p_{\theta_D}(R^S_{hw})) \tag{19}$$

$$-\sum_{I_T \in T}\sum_{h \in H}\sum_{w \in W} \log(1 - p_{\theta_D}(R^T_{hw})) \tag{20}$$

  where $R^S_{hw} = \phi_{l-1}(\theta, I_S)_{hw}$ and $R^T_{hw} = \phi_{l-1}(\theta, I_T)_{hw}$ denote source and target representation of each units, respectively

- inverse domain loss for convenience:

$$\mathcal{L}_{\text{Dinv}} = -\sum_{I_S \in S}\sum_{h \in H}\sum_{w \in W} \log(1 - p_{\theta_D}(R^S_{hw})) \tag{21}$$

$$-\sum_{I_T \in T}\sum_{h \in H}\sum_{w \in W} \log(p_{\theta_D}(R^T_{hw})) \tag{22}$$

$$\tag{23}$$

- alternating minimization procedure

$$\begin{array}{ll} \min\limits_{\theta_D} & \mathcal{L}_D \\ \min\limits_{\theta} & \frac{1}{2}[\mathcal{L}_D + \mathcal{L}_{\text{Dinv}}] \end{array}$$

- optimizing these two objectives iteratively amounts to learning the best possible domain classifier for relevant image regions and then using the loss of that domain classifier to inform the training of the image representations so as to minimize the distance between source and target domains

5.5 Category Specific Adaptation

- **TODO: cite 25 26?**
- compute by per image labeling statistics in source domain P_{L_S}. for each source image which contains class c, compute percentage of image pixels which have a ground truth label corresponding to this class. Can then compute histogram over these percentages and denote the lower 10% boundary as α_c, average values as δ_c and upper 10% as γ_c

- use this to inform target domain size constraints, explicitly transferring scene layout information from source to target domain (e.g. street usually takes much more space in a autonomous driving image than signs).
- constrained MIL for the case where image-level labels are known: for a given target image for which a certain class c is present, impose following constraints on output prediction map $p = \arg\max \phi(\theta, I_T)$

$$\delta_c \leq \sum_{h,w} p_{hw}(c) \leq \gamma_c \tag{24}$$

- encourages to not assign more pixels to class c than expected range observed in source domain
- optimize this objective with lower bound slack to allow for outlier cases wehre c simply occupies less of the image than is average in source domain. Do not allow slack on upper bound constraint as it is important that no single class occupies too much of any given image
- general constraint: can be applied to all classes
- optimize as in Pathak et al **TODO: cite 25**
- modification: to prevent model diverging and over-fitting: use size constraint that if the lower 10% of source class distribution α_c is greater than 0.1, the down-weight the gradients due to these classes by factor of 0.1. Can be viewed as re-sampling of the classes so as to come closer to a balanced set, allowing the relatively small classes potential to inform the learning objective
- need image-level labels for this approach, thus complete approach: predicting image-level labels, then optimizing for pixel predictions that satisfy the source transferred class size constraints. (e.g. source semantic segmentation yields: x pixels street, y pixels car, z pixels signs. target image-level labels are street and signs, make sure that semantic segmentation has at max x street and z sign classified pixels)
- **TODO: reread last paragraph before experiments**
- given target image I_T, compute current output class prediction map $p = \arg\max \phi(\theta, I_T)$
- for each class comput percentage of pixels assigned to it in current prediction $d_c = \frac{1}{H \cdot W} \sum_{h \in H} \sum_{w \in W} (p_{hw} = c)$
- assign image-level label to class c if $d_c > 0.1 \cdot \alpha_c$ (i.e. if currently labeling at least as many pixels as 10% of the expected number for a true class appearing in the image)

5.6 Experiments

domain adaptation tasks:

- cities2cities
- season2season
- synth2real
- use front-end dilated FCN (**TODO: cite 33**) as both the initialization for our method and as baseline model for comparison
- all code and models trained and evaluated in the Caffe (**TODO: cite 16**) framework, available before camera-ready
- use IoU
- for c2c and syn2r follow evaluation protocol of **TODO: cite 3** and train models with 19 semantic labels of Cityscapes
- for s2s use 13 semantic labels of SYNTHIA instead

5.7 Datasets

5.7.1 Cityscapes

- 34 categories
- high resolution 2048×1024
- three parts: 2975 training samples, 500 validation samples, 1525 test samples
- split into european cities, different geographic and population distributions

5.7.2 SYNTHIA

- 13 classes with different scenarios and sub-conditions
- for season to season use SYNTHIA-VIDEO-SEQUENCES, different cities, seasons, weathers, illumination, captured by 8 RGB cameras 360° visual field, use only dash-cam for this paper
- for synth2real use SYNTHIA-RAND-CITYSCAPES, 9000 random images from all sequences with Cityscapes-compatible annotations, as source domain data

5.7.3 GTA5

- 24966 high quality labeled frames from realistic open-world computer game Grand Theft Auto V (GTA5)
- frames with 1914×1052 generated from fictional city Los Santos based on Los Angeles, CA.

- use whole dataset with labels compatible to Cityscapes categories for synth2real adaptation

5.7.4 BDDS

- thousands of densely annotated dashcam video frames, hundreds of thousands of unlabeled frames
- 1280×720, 34 categories compatible to Cityscapes label space
- majority of data from New York and San Francisco (representative for eastern and western coasts)
- different from other existing driving dataset covers more challenging conditions (urban streetview at night, highway scene in rain,..)

5.8 Quantitative and Qualitative Results

- first large distribution shift (synth2real)
- then medium shift (season2season)
- small shift (city2city in Cityscapes)

TODO: add import table contents

5.8.1 Large Shift: Synthetic to Real Adaptation

- for GTA to Cityscapes: compared to performance of source dilation model the domain adversarial training contributes 4.4% raw and $\sim$ 20% relative percentage mIoU improvement and multiple instance loss another 1.6% raw and $\sim$ 6% relative
- for SYNTHIA to Cityscapes also measurable improvement
- raw 0.9% for adversarial, raw 1.5% with additional multiple instance loss

5.8.2 Medium Shift: Cross Seasons Adaptation

- SYNTHIA season2season
- on average $\sim$ 3% mIoU improvement. Higher mIoU for 12/13 categories
- no improvement on class *car* (probably because cars have little to no change in appearance through seasons in SYNTHIA)
- largest performance improvements through this method on categories like *road* in shift from fall to winter

5.8.3 Small Shift: Cross City Adaptation

- raw 3.6% mIoU improvement through domain adversarial training, only 0.1% through multiple instance loss
- only noticeable improvement from category specific alignment on *traffic light, rider, train*, probably because domain shift between train and val sets mainly results from change in global appearance due to difference in city, specific category appearance may not change significantly though

5.9 BDDS Adaptation

TODO: find quantitative results

6 From Virtual to Real World Visual Perception using Domain Adaptation - The DPM Example

see [**?**]

6.1 Need for Virtual Worlds

- in general best performing machine learning algorithms are *supervised* i.e requiring annotated information (ground truth) for training
- human annotators (e.g. Amazon Mechanical Turk, LabelMe **TODO: cite 48 and 53**) used for semantic segmentation but can't for pixel-wise optical flow and depth
- ImageNet Large-Scale Visual Recognition Challenge (ILSVRC)
- ImageNet contains over 15 million of humal-labeled (using Mechanical Turk) high-resolution images of roughly 22,000 categories. (gigantic human annotation effort)
- ILSVRC uses subset thereof containing 1k images of 1k categories, 1.2M for training, 50k for validation, 150k for testing
- many deep CNNs developed today rely on an ImageNet pre-trained deep CNN which is modified or fine-tuned to solve a new task or operate in a new domain
- key for success of research community are powerful GPU hardware and large dataset with ground truth
- Cityscapes dataset: driving through 50 cities covering several months and weather conditions

- getting ground truth like in Cityscapes can take 30 to 90 minutes per image for a human in case of fine-grained annotations and depending on image content.
- pixel-wise optical flow and depth are very hard or impossible to obtain by human annotators without active sensors
- work on using realistic virtual worlds for training vision-based perception modules roughly since 2008 due to difficulty of optaining and relevance of having large amounts of data with ground truth for training, debugging and testing
- advantages:
  1. forcing driving and data acquisition situations needed
  2. optaining different types of pixel-wise ground truth (class ID, instance ID, depth, optical flow)
  3. generating such data relatively fast (SYNTHIA environment can generate 10k images per hour with such ground truths using standard consumer hardware)
- questions of such a proposal were:
  - can a visual model learned on virtual worlds perform well in real-world environments?
  - does this depend on the degree of photo-realism?
- **TODO: reread last paragraph before Need for DA (page 4) for citations and examples of other syn2real proposals**

6.2 Need for Domain Adaptation

- problem of domain adaptation is not just a virtual-to-real issue but rather a sensor-to-sensor or environment-to-environment one **TODO: cite 70, 71**
- citations 32, 33, 63, 43 for other syn2real DA
- **TODO: HOG+LPB/Linear-SVM paradigm 68, Haar+EOH/AdaBoost 69**
- **TODO: deformable part-based model DPM 17**
- focused on supervised domain adaptation in most cases (relatively few amount of annotated target-domain data used to adapt the model learned with source-domain data)
- for holistic models focused on mixing source and target data collected via active learning for model adaptation (term corresponding feature space as *cool world*) (*mixing source and target data in the mini-batches*)

- for DPM focus on using just source-domain model together with target-domain data (*fine-tuning*)
- DPM was state of the art before breakthrough of deep CNNs
- being based on HOG-style features, training data in order of a few thousands is sufficient as more data doesn not really translate to better accuracy **TODO: cite 81**
- DPM can be reformulated as a deep CNN for end-to-end learning
- use own DA method for DPM termed as *Structureaware Adaptive Structural SVM* (SA-SSVM)

6.3 Domain Adaptation for DPM in a Nutshell

- DPM encodes the *appearance* of objects' constituent *parts* together with a *holistic* object representation termed as *root*.
- allows parts to be located at different positions with respect to the root
- plausible relative locations (*deformations*) are also encoded
- both appearance and deformations are learned
- appearance of parts is learned at double resolution than the root
- the triplet root-parts-deformations is known as *component*
- to avoid too blurred models, DPM allows to learn a mixture of components
- different components correspond to very different object views or poses, specially when this implies very different aspect ratios of corresponding root Bounding Box (BB)
- in practice a DPM is encoded as a vector $\mathbf{w}$ which has to be learned
- let $\mathbf{w}^S$ model learned with source-domain data
- SA-SSVM DA method takes $\mathbf{w}^S$ and relatively few target-domain data to learn new $\mathbf{w}$ model which is expected to perform better in the target domain
- **TODO: see citation 77 for mathematical technical details of SA-SSVM**
- see Fig. 3 page 6 for reference of following explanations
- $\mathbf{w}^S$ consists of components: half body and full body, as well as persons seen from different viewpoints
- each component consists of root and parts (head, torso, etc.)
- to adapt this DPM to a target domain, decompose it as $\mathbf{w}^S = [\mathbf{w}_1^{S'}, ..., \mathbf{w}_P^{S'}]$ where P is number of structures and u' stands for transpose of u

- each component $\mathbf{w}_p^S$ may contain both appearance and deformation parameters (for roots only appearance)
- the decomposed model parameters are adapted to the target domain by different weights, denoted by $\beta_p, \quad p \in \{1, P\}$
- i.e. the SA-SSVM procedure allows domain adaptation for each of such structures seperately by defining $\Delta\mathbf{w}_p = \mathbf{w}_p - \beta_p \mathbf{w}_p^S, \quad p \in \{1, P\}$
- in order to learn these adaptation weights introcude regularization term $\|\boldsymbol{\beta}\|^2$ in the objective function where $\boldsymbol{\beta} = [\beta_1, ..., \beta_P]'$
- and use a scalar parameter γ to control its relative penalty
- Finally, C and ξ_i are just the standard terms of a SVM objective function and N the number of target-domain samples used for the adaptation
- after optimizing the objective function, $\mathbf{w} = [\mathbf{w}_1', ..., \mathbf{w}_P']'$ is the domain adapted DPM