Reiko Lettmoden
r.lettmoden@tu-bs.de

Supervisor Maximilian Menke
maximilian.menke@de.bosch.com
Robert Bosch Car Multimedia GmbH

Referee Prof. Dr.-Ing. Martin Eisemann
eisemann@cg.tu-bs.de
Computer Graphics Lab, TU Braunschweig

Co-Referee Prof. Dr.-Ing. Andreas Schwung

Automatic patch sampling for GANs through image retrieval

Bachelor Thesis

April 29, 2022

Computer Graphics Lab, TU Braunschweig

Eidesstattliche Erklärung

Hiermit erkläre ich an Eides statt, dass ich die vorliegende Arbeit selbständig verfasst und keine anderen als die angegebenen Hilfsmittel verwendet habe.

Braunschweig, 29. April 2022

Reiko Lettmoden

Zusammenfassung

Neuronale Netze werden im autonomen Fahren weitflächig genutzt und benötigen große Mengen an Daten. Synthetische Daten können von Simulationen generiert werden, während das Labeln der Daten im Vergleich zu echten Daten weniger menschliche Arbeit benötigt. In Domain Adaption (DA) wird das Wissen einer gelabelten Source-Domäne auf eine ungelabelte Ziel-Domäne adaptiert, auf der eine Aufgabe ausgeführt wird, in diesem Fall Fahrzeug-Erkennung.

Style-Transfer GANs werden in DA benutzt, um die Source-Domäne mit der Ziel-Domäne anzupassen, sodass die Aufgabe auf style-transferrierten Bildern gelernt wird. Bildausschnitte werden während des GAN Trainings zufällig ausgeschnitten, wodurch der GAN Artifakte generiert [RAHK22]. Richter et. al [RAHK22] schlagen vor, beide Domänen mit passenden Bildausschnitten aneinander anzupassen, wodurch weniger Artifakte generiert und Texturen beider Domänen besser aneinander angepasst werden. Diese Bachelorarbeit untersucht, ob das Samplen von Bildausschnitten durch das Verbessern von Style-Transfer Metriken einer DA Aufgabe verbessern kann.

Als DA Aufgabe wird Fahrzeug-Erkennung vom synthetischen Datensatz *Driving in the Matrix* [JRBM+17] zum reellen Datensatz *Cityscapes* [COR+16] adaptiert. Das von Richter et al. [RAHK22] implementierte Samplingframework wird erweitert, um gematchte Bildausschnitte mit semantischen Segmentierungsmasken zu verifizieren und gewichten. Metriken zur perzeptuellen und hohen semantischen Evaluation von Style-Transfer werden diskutiert, die mit der Leistung der hintergelagerten DA Aufgabe korrelieren. Experimente in der Arbeit zeigen, dass das Matchen mit VGG16[SZ15] Features Style-Transfer und mAP verbessert, da Texturen besser angepasst werden. Experimente mit hoher semantischer Verifikation und einem Fahrzeug spezifischen Samplingalgorithmus führen dazu, dass GANs degenerieren und somit zu einer niedrigeren mAP. Mehr Bildausschnitte zu samplen führt zu besseren Matches und größeren Datensätzen, was Style-Transfer verbessert. Das Hochgewichten von Bildausschnitten mit Fahrzeugen führt dazu, dass der GAN Halluzinationen in Fahrzeugen generiert, was mAP verbessern kann. Die vorherige Vergleichslinie basierend auf einem CycleGAN [ZPIE17] mit zufälligem Sampling erzielt 46.61 mAP, während die Standardkonfiguration der Arbeit 48.05 mAP und beste Konfiguration 49.20 mAP erzielt.

Abstract

Neural networks are widely used in autonomous driving and require data in great quantities. Synthetic data can be generated by simulations while labeling data takes less human work in comparison to real data. In domain adaption, knowledge from a labeled source dataset is adapted to an unlabeled target domain on which a task is performed, in this case vehicle detection.

Style-transfer GANs are used in domain adaption to align the source domain with the target domain, such that a downstream task can be learned on style-transferred images. Patches are randomly cropped during GAN training which causes the GAN to generate artifacts[RAHK22]. Richter et. al [RAHK22] propose to align both domains by matching patches which causes less artifacts and improves textural domain alignment. Hence, this thesis examines whether sampling patches for GAN training can improve domain adaption downstream task performance by improving style-transfer.

As domain adaption task, vehicle detection is adapted from the synthetic dataset *Driving in the Matrix* [JRBM+17] to the real dataset *Cityscapes* [COR+16]. The provided patch sampling framework from Richter et al. [RAHK22] is enhanced to utilize semantic segmentation masks to verify and weight matches. Metrics for perceptual and high semantic style-transfer evaluation are discussed which correlate with downstream task performance. Experiments in this thesis show that matching patches with VGG16[SZ15] features improves style-transfer and mAP since textures are aligned better. Experiments with high semantic verification and a vehicle specific sampling algorithm caused the GANs to degenerate and thus worsened mAP. Sampling more patches results in better matches and larger datasets which improves style-transfer. Up-weighting matches containing vehicles causes the GAN to generate minor hallucinations in vehicles which can improve mAP. The previous style-transfer baseline based on a CycleGAN[ZPIE17] with random sampling achieves 46.71 mAP, while our default configuration holds 48.05 mAP and the best configuration reaches 49.20 mAP.

Contents

Chapter 1

Introduction

Autonomous driving has drawn much attention to improve the driving experience. Difficult tasks, such as object detection and semantic segmentation, must be performed in real-time with sufficient performance. Problems in computer vision are complex since the scene can change through occlusion, scale, translation and illumination while the content remains mostly the same. Neural networks achieve state-of-the-art results in these tasks since they can adapt to these scene changes conditions[LLC+21][RAHK22]. Contrary to handcrafted algorithms, neural networks are trained with data for a task and provide a prediction for a given input depending on their parameters. In supervised learning, labels are given which hold the ground-truth. A loss is calculated with the objective to minimize the error between prediction and the ground-truth.The loss is then backpropagated through the network, adjusting parameters of the network to improve predictions. Hence, supervised neural networks in computer vision rely on images with labels depending on the tasks. For example, object detection requires multiple labeled object instances containing several object classes within an image. As downstream tasks in computer vision are complex and the input space is high compared to other domains, neural network have millions of parameters.

Training a neural network therefore requires high amounts of data, which needs to be labeled manually by human annotators. Labeling data takes time and can become expensive for large datasets. On the other hand, synthetic data rendered from simulations such as GTA V[RVRK16][JRBM+17] can be generated in great quantities while the labels can be derived from the simulation engine almost automatically. The task of transferring knowledge of one labeled domain to another unlabeled domain is called domain adaption (DA). The difference in content and style between the synthetic and real domain is referred to as domain gap. Since the domain gap causes neural networks to overfit to the synthetic domain, knowledge can not be transferred to the target domain. To reduce the domain gap, one approach

is to transfer the style of the synthetic data to the real data, thus aligning both data distributions on the image level.

Generative adversarial networks (GANs)[GPAM+14] achieve state-of-the-art performance in style transfer, since they are also able to modify textures in comparison to color transfer and photo style-transfer [RAHK22]. GANs consists of a generator and adversarial discriminator, which play a minimax game [GPAM+14]. The generator tries to recreate a target distribution from a latent space while the discriminator decides whether the generated input is from the real or generated distribution. Style-transfer GANs suffer from artifacts, e.g. layout differences of both domains are adjusted incorrectly with deformations in style or content. For example, if the target domain is more likely to have trees in the upper half of the image, where for the source domains is more likely sky instead, the GAN learns to put trees in the sky. Richter et al. [RAHK22] assume, that the artifacts are caused by cropping patches randomly for GAN training on both domains, which was done before [HTP+18]. Since domain specific layout is sampled by random cropping, [RAHK22] improve their approach by introducing a patch sampling framework. Patches are sampled on both data domains and features extracted with a VGG16[SZ15] network. Thus, patches are matched and weighted such that every region in the source image is evenly sampled.

This thesis studies patch sampling for style-transfer GANs to improve domain adaption downstream task performance. Therefore, methods from image retrieval are applied to improve the baseline from Richter et. al [RAHK22] on a vehicle detection downstream task. Matching is enhanced with semantic verification to filter out mismatches and matching on a high semantic feature space is compared with a search on perceptual VGG16[SZ15] features. The influence of a domain bias shift on style-transfer by up-weighting or sampling more vehicle matches is investigated. Additionally, patch sampling is examined with reference to the matching quality, number of matches and patch size. Metrics are discussed to evaluate style-transfer both perceptually and on high semantics.

In the following, a structural overview of this thesis is given. Chapter 2 discusses recent advances in image retrieval, image recognition and domain adaption. Chapter 3 introduces methods for patch sampling, and discusses metrics to evaluate style-transfer. A framework to sample patches offline and training configurations is presented in chapter 4. Style-transfer experiments are visualized and evaluated in chapter 5, while chapter 6 discusses possible future work and results from the experiments. Additional visualizations of sampled matches can be found in the appendix 7.

Chapter 2

Related Work

Generative Adversarial Networks Goodfellow et al. [GPAM+14] first introduced generative adversarial networks (GANs), which describe a framework of a generator and a discriminator. The generator maps an input from a randomly generated latent space to a target distribution. The adversarial discriminator on the other hand distinguishes from an input, whether it is real or is generated by the generator. Both play a min-max game in which the generator tries to make the discriminator assign the wrong label by generating samples from the real data distribution. On the other hand, the discriminator tries to learn assigning the correct label to real and fake data.

In the case of style-transfer GANs, an image is fed into the generator and the style is transferred to the target style, while the content is preserved. To improve semantic consistency and avoid GAN degeneration during training, further constraints stabilize GAN training through enforcing cycle consistency [ZPIE17]. Cycle consistency states that generated images should be able to be mapped back to the input. CyCADA [HTP+18] trains a style-transfer GAN based on the proposed CycleGAN [ZPIE17] model with a cycle consistency loss and an additional downstream task loss [HTP+18]. CyCADA samples patches randomly during training, e.g. an image is randomly chosen and a random patch of a fixed size is cropped for the source and target domain respectively. For this thesis, CycleGAN[ZPIE17] is applied for style-transfer.

Domain Adaption Domain adaption (DA) is a task where knowledge from a source domain is utilized to learn a task on the target domain, such as object detection or semantic segmentation. Since real data is expensive and synthetic data can be attained easily in comparison, DA is often performed from a synthetic dataset generated by a photorealistic simulation to a real world dataset. Thus, for semantic segmentation, DA is performed from *Playing for Data* [RVRK16] to *Cityscapes* [COR+16], whereas for ve-

hicle detection *Driving in the Matrix* [JRBM+17] is adapted to *Cityscapes*. To overcome the domain gap, both domains can be aligned through adversarial training or self-training by generating pseudo labels on the target domain[HDVG21]. This thesis studies adversarial domain alignment, which is done through style-transfer as proposed by CyCADA [HTP+18]. ProDA [ZZZ+21] is used to generate semantic segmentation pseudo labels on both domains. During this thesis, DAFormer [HDVG21] was published which outperforms ProDA by far and proposes a rare class sampling to increase mIoU on rare classes. This thesis adapts [HDVG21] approach of rare class sampling to patch-sampling and thus improve the variety of sampled-patches. [LKL+20] align both domains by searching the nearest neighbor on the target domain based on semantic pseudo labels. As domains are aligned on image level, this approach is adapted for patch matching to filter out semantic mismatches.

Image Recognition In image recognition, features are extracted from an image by a neural network to perform a downstream task, e.g. classification or object detection. Therefore, the network is trained on the downstream task to optimize parameters of the network. The neural network architectures are compared on classification datasets.

Previously, convolutional neural networks (CNNs) dominated the field of image recognition by stacking convolutions onto multiple layers [KSH12]. This way, features are learned hierarchically, e.g. color and edge filters are learned in the first layers, textures in the middle layers and in the higher layers semantic properties of classes [ZF14]. Because deep neural networks are unable to learn at the later stage of training due to the vanishing gradient, [HZRS16] introduce skip connections. Their proposed network for image recognition enhances a CNN with skip connections, which is then termed as a residual network (ResNet)[HZRS16]. Visual transformers (ViT) were proposed by Dosovitsky et al. [DBK+21] and thus introduced self-attention to image recognition. Transformers learn spatial relations through whereas CNNs have a strong local bias due to convolutions[DBK+21].

Currently, transformer and CNNs are in a head-to-head race for best performance. While Swin-Transformer[LLC+21] reaches state-of-the-art performance for the transformer architecture, ConvNeXt[LMW+22] was published recently, which proposes architectural and training improvements for CNNs. Both provide state-of-the-art performance while requiring less compute power than the Swin-Transformer. VGG16 [SZ15] is a CNN with 16 layers and is often used to extract perceptual feature vectors [WLZ+18] [YUC+19] [ZIE+18]. Richter et al. [RAHK22] extract and match patches with VGG16 features, such that VGG16 is also used in this thesis to match patches.

Image Retrieval In image retrieval, an image is queried to a database such that the most similar images based on their content are retrieved. Methods for image description [CAS20][NBTM20], storage [NAS+17][CAS20] and querying [NAS+17][RTC18] are developed to provide fast image retrieval while achieving good retrieval results.

Feature vectors are used to describe image patches. Though visual transformers are not used yet in image retrieval, the ResNet is used in recent image retrieval models such as SOLAR [NBTM20] and the state-of-the-art DELG [CAS20] model. Image retrieval models take advantage of the hierarchical architecture of CNNs, since local features are extracted from the conv_4 layer whereas global features are extracted from the conv_5 layer [CAS20] [TJC20]. DELG utilizes global features for a global image search and then performs verification on local features. For this thesis, only local features are relevant, since global features neglect local context. VGG16 has a receptive field of 196 pixels and thus is smaller than the ResNet50 conv_4 and conv_5 receptive field of the size 267 and 427 respectively. Thus, the ViT, ResNet conv_4 and VGG can be used for patch description. To attain more distinct feature vectors, either pooling can be applied or feature extractors optimized by fine-tuning. For fine-tuning, pre-trained networks are trained on a dataset from the retrieval domain [NAS+17], for example on a classification task [CAS20]. Another approach is to fine-tune with a contrastive loss, which encourages positive feature vectors to be similar and negative feature vectors to be different [RTC18]. Positive and negative feature vectors are weakly labeled based on GPS positions [AGT+16] or a 3D reconstruction through structure-from-motion [RTC18]. This requires collecting datasets which could be done in future work. Feature maps extracted from a CNN can also be re-weighted based on attention scores to up-weight relevant and down-weight irrelevant feature descriptors [NAS+17]. SOLAR [NBTM20] re-weight feature maps based on self-attention, whereas DELG learn two convolutional layers to create attention scores[CAS20].

Pooling is applied on the extracted feature maps which causes the feature vectors to be more distinctive. Sum pooling of convolutions (SPoC)[BL15] can be adapted to average pooling, whereas maximum of convolutions (MAC) pools the channel-wise maximum of the feature map, which can also be applied regionally (R-MAC)[TSJ16]. [RTC18] introduce the generalized mean (GeM) which is learned end-to-end and can be adapted to SPoC or MAC. DELG [CAS20] apply GeM with a fixed parameter which is evaluated in this thesis.

In image retrieval, the dimensionality of computed feature vectors can be reduced by principal component analysis (PCA)[WEG87] or by learning an auto-encoder end-to-end [CAS20]. [JC12] show that PCA can even improve retrieval results if it is applied with whitening to avoid overcounting co-occurrences. Afterwards, product quantization can be applied to allow efficient approximate nearest neighbor search, while further reducing required

storage space. As sampling performance benefits from a larger search space, product quantization (PQ)[JDS10] and PCA can be applied if faster search or less space is required. PCA and PQ are implemented by FAISS library [JDJ19], which allows to run these algorithms efficiently on GPUs.

Metrics The results of GAN trainings are evaluated perceptually and on the downstream task performance. For perceptual evaluation, the KID [BSAG18] describes the squared MMD [GBR+12] of InceptionV3 [SVI+16] features from style-transferred images and real images. [RAHK22] state, that the KID is not able to measure perceived realism well and thus propose the sKVD_i, which calculates the squared MMD [GBR+12] of features from the i-th convolutional layer of the VGG network instead. Contrary to the KID, [RAHK22] sample for style-transferred patches the semantic nearest neighbor in the target dataset and then calculate the sKVD_i. Hence, the sKVD_i is best suited for perceptual evaluation of style-transfer and thus is applied in this thesis.

The VOC challenge [EVGW+10] propose the mIoU (mean intersection over union) for semantic evaluation. For object detection, the mAP (mean average precision) as proposed in the *MS COCO* dataset[LMB+14] is evaluated.

Chapter 3

Patch Sampling for Style-Transfer GANs

Style-Transfer GANs for domain adaption are trained with patches of both domains, such that the generator learns to transfer the style of images from the source domain to the style of the target domain. Let P_S be a set of patches from the source domain $D_S = \{I_S^1, I_S^2, ..., I_S^n\}$, where n is the number of images in the source domain, P_T respectively from the target domain $D_T = \{I_T^1, I_T^2, ..., I_T^m\}$. For every image I_S^j from the source domain, k patches $P_S^{j,1}, P_S^{j,2}, ..., P_S^{j,m} \subset I_S^j$ are cropped, such that $k \times n$ crops are sampled from the source domain. Analog, l patches are sampled per image I_T^j for the target domain and $l \times m$ crops in total respectively. Both image datasets are unpaired, e.g. both datasets have a domain specific layout and content, such that cropping and matching patches randomly causes the discriminator to learn distinguishing both domains by artifacts. Hence, the generator learns to generate layout artifacts, which influence downstream task performance negatively. Aligning patch distributions causes the discriminator and generator to focus on domain specific features such as textures of objects[RAHK22]. Thus, a matching $M \subset P_S \times P_T$ of patches from both domains is constructed for GAN training. Therefore, sampling, matching and weighting strategies are discussed which can improve style-transfer for domain adaption.

3.1 Matching

Richter et al. [RAHK22] map sampled patches to VGG16[SZ15] feature space ϕ_{VGG} pre-trained on ImageNet[DDS+09] and match a patch $p_i \in P_S$ with $p_j \in P_T$, if the cosine similarity is above 0.5, as shown in equation 3.1. This is the baseline for matching, as no patch sampling was performed

before.

$$P_{match}(p_i) = \left\{ p_j \in P_T \middle| \frac{\phi_{\text{VGG}}(p_i) \cdot \phi_{\text{VGG}}(p_j)}{\|\phi_{\text{VGG}}(p_i)\| \, \|\phi_{\text{VGG}}(p_j)\|} > 0.5 \right\} \tag{3.1}$$

The receptive field of the last VGG16 ReLU activation at the last convolutional layer is 196 pixels[RAHK22], describing the size of a region that influences the feature vector of a layer. If the patch size for sampling is smaller than the receptive field, features are extracted with a patch size of 196 and afterwards center-cropped to the original patch size. If the patch size is larger than the receptive field, patches can be resized to the receptive field size or can be generalized mean (GeM) pooled [RTC18], which is shown in equation 3.2.

$$f_k = \Big(\frac{1}{|X_k|} \sum_{x \in X_k} x^p \Big)^p \tag{3.2}$$

Let X be the 3D feature map $\mathbb{R}^{W_f \times H_f \times K_f}$ after a convolutional layer, such that $W_f \times H_f$ feature vectors are extracted of the dimension K [RTC18]. Then, k iterates over the k-th dimension for all feature vectors from the feature map. According to current work in image retrieval from Cao et al. [CAS20], p is set to 3. GeM puts emphasis on dominant features in patches since minor activations decrease through pooling.

Zeiler et al. [ZF14] visualize CNNs with activation maps from convolutional filters. They show that filters in the first layers learn color and gradient filters. Further, mid layers learn based on textures and higher layers are activated by class properties or class instances, which is visualized in figure 3.1.

Matching patches by a mapping based on ϕ_{VGG} aims to align both distributions, but mismatches occur, because patches are matched due to textures or a single object occurring in the patch, for example signs are matched (see figure 3.2). Due to mismatches, the GAN may learn to style-transfer object classes semantically incorrect.

To filter out mismatches, high-level semantic information from semantic segmentation can be used. Semantic segmentation is a task in computer vision where each pixel in an image is assigned a class c such as street or car from a set of classes C. As no labels may be used in the target domain, semantic segmentation is a challenging task in domain adaption. ProDA[ZZZ+21] is used as a recent model to extract semantic segmentation masks L^S_{Pseudo} offline for images on the source domain and L^T_{Pseudo} respectively on the target domain. Patches are mapped to a semantic probability density function (in the following denoted by semantic *pdf*) ϕ_{SEM} for each class, as described in eq 3.3.

$$\phi_{\text{SEM},i}(P_{\text{Pseudo}}) = \frac{\sum_j [P_{\text{Pseudo},j} = c_i]}{\text{H}_P \cdot \text{W}_P} \tag{3.3}$$

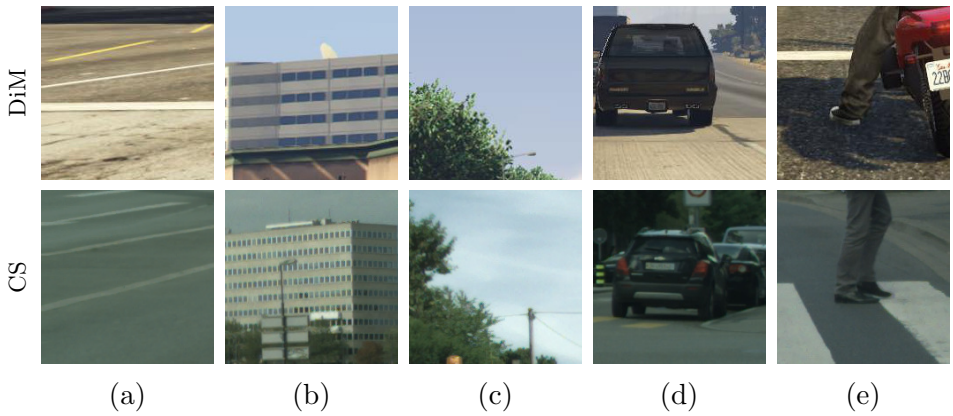


Figure 3.1: Matching patches from the GTA dataset *Driving in the Matrix* (DiM) [JRBM+17] with *Cityscapes* [COR+16] patches based on VGG16 [SZ15] features. Texture based matches can be seen in (a)-(b), whereas (c) shows a matching based on classes and scene layout. (d) is matched due to the class car and (e) shows a matching based on class properties, e.g. legs.

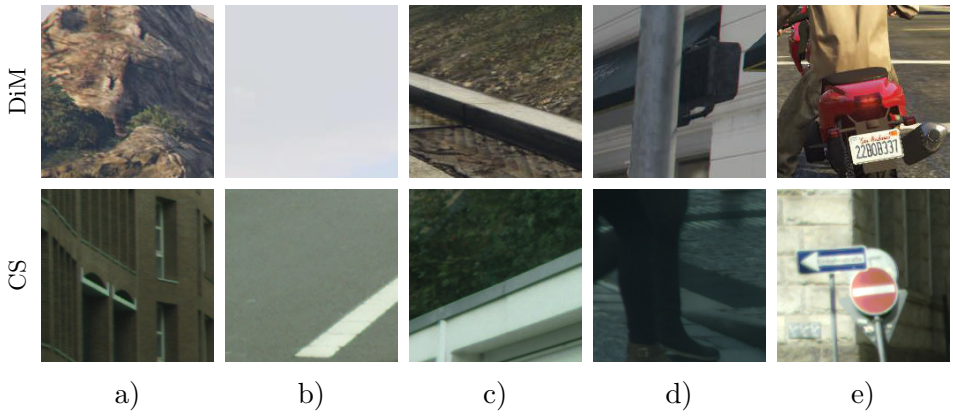


Figure 3.2: Mismatches of patches based on VGG16 [SZ15] features. Mismatches due to textures (a)-(b) represent, whereas (c)-(d) are matches due to scene layout and e due to ImageNet classes.

Therefore, for every pixel j in patch P_{Pseudo}, the class occurrences are counted for every class $c_i \in C$ through the Inversion-Bracket $[\cdot]$. The Inversion-Bracket returns 1 if a condition is true and else 0. The class occurrences are then normalized by patch size $H_P \times W_P$ 3.3. Similar to eq. 3.1 vectors are matched, if the cosine similarity is above a threshold t_{SEM} as seen in eq. 3.4.

$$P_{\text{match\_verified}}(p_i) = \left\{ p_j \in P_T^{Pseudo} \middle| \frac{\phi_{\text{SEM}}(p_i) \cdot \phi_{\text{SEM}}(p_j)}{\|\phi_{\text{SEM}}(p_i)\| \, \|\phi_{\text{SEM}}(p_j)\|} \geq t_{\text{SEM}} \right\} \quad (3.4)$$

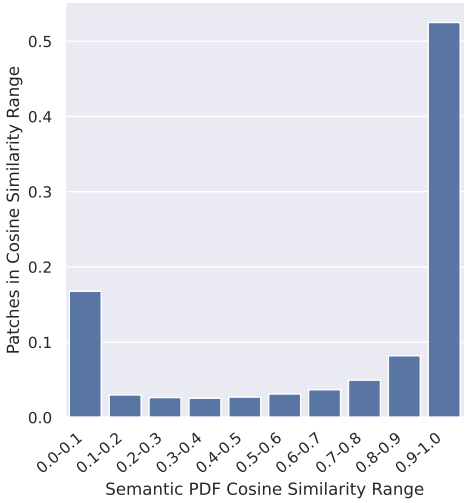


Figure 3.3: Distribution of matches according to their semantic *pdf* similarity. Matches are fulfilling eq. 3.1 with the patch size 196 being evaluated.

Figure 3.3 shows that approximately a quarter of the matches found by searching in ϕ_{VGG} have a semantic *pdf* similarity below 0.5, with most mismatches being in the interval $[0; 0.1]$. Therefore, expanding the search with a semantic verification can filter out mismatches and align matches further with high semantics.

Additionally, experiments are evaluated with a search on semantic segmentation feature vectors, such that ϕ_{VGG} is replaced with ϕ_{SEM} in eq. 3.1. Matches are calculated according to Richter et al. [RAHK22], who match k nearest neighbors with FAISS[JDJ19] from the source domain with the target domain. Hence, matches are pre-computed efficiently and loaded during training of the style-transfer GAN.

3.2 Sampling

Style-transfer GANs are influenced by sampling and the patch sizes, which is discussed this chapter. Further, sampling sizes on the target influences the matching and dataset diversity, whereas sampling algorithms shift the dataset distribution.

3.2.1 Domain Adaptive Sampling

Richter et al. [RAHK22] randomly sample square patches on both the synthetic and real data domain. Since [RAHK22] aim to generate realistic images without performing any domain adaption (DA) task, random sampling causes the GAN to stylize common classes well because the domain specific

distribution of content is preserved. However, random sampling causes DA relevant classes to be sampled rarely, as seen in figure 3.4. Therefore, sampling more patches of DA classes may improve the capabilities of the GAN to stylize those classes better and thus object detection results.

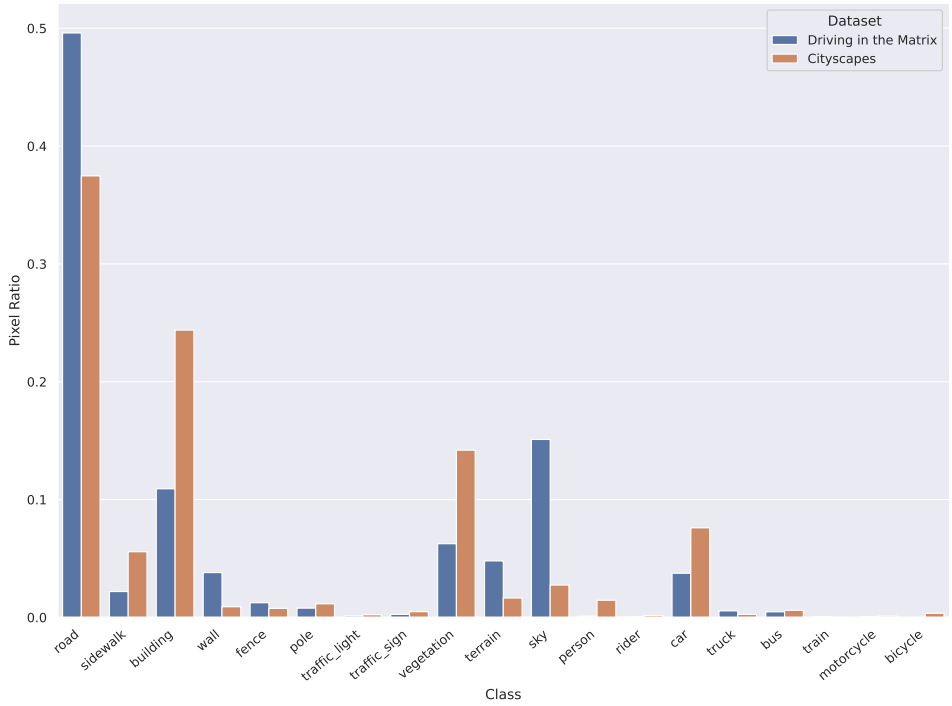


Figure 3.4: Class distribution of patches sampled randomly from *Cityscapes* (CS) and *Driving in the Matrix* (DiM). Domain adaption relevant classes (car, truck and bus) make up approximately 5% of all pixels on the source dataset.

The proposed DA sampling requires semantic segmentation masks to find patches containing a class. Since DiM is a vehicle detection data set and no labels may be used in the target domain, semantic pseudo labels L^T_{Pseudo} and L^S_{Pseudo} are pre-computed with ProDA [ZZZ+21] for all images. Approximately half of the sampled patches should contain DA relevant class pixels for the proposed DA sampling, while the rest is sampled randomly to avoid overfitting of the GAN and support stable style-transfer.

3.2.2 Patch Size

The patch size determines, whether more local or global information is sampled and thus influences the training of the GAN. Since larger patch sizes yield the same content but not the same layout, the discriminator learns to distinguish between both domains by layout differences[RAHK22]. Additionally, the generator stylizes larger regions, which is harder to stabilize.

On the other side, small patch sizes lead to more local content being sampled, e.g. a patch is more likely to contain one class. Thus, the GAN adapts less with a smaller patch size and causes less artifacts, which on the other hand also decreases the capability of style-transfer.

3.2.3 Sampling Size

The sampling size describes the number of crops per image on the target and source domain, e.g. the parameters k and l. Higher sampling sizes on the target domain influences the quality of matching since more samples increase the search space for matching. Similar to [SSSG17], more data leads to logarithmic increase of matching similarity. Thus, experiments are evaluated to show a good tradeoff between performance and computation time. Cropping more images on the source domain increases the diversity of the patches such that the style-transfer GAN can generalize well. Optimally, the GAN receives for every image a distinct patch each epoch.

3.3 Weighting

After matching, image regions on the source domain are not covered uniformly due to patch sampling3.2.1 and matching3.1. Random sampling causes regions at the edges to be sampled less frequently in comparison to inner image regions since patches overlap less on edges. Sampled source patches are discarded during matching because no match can be found.Thus some image regions in the source domain are cropped more often in comparison to others, which can be seen in figure 3.5.

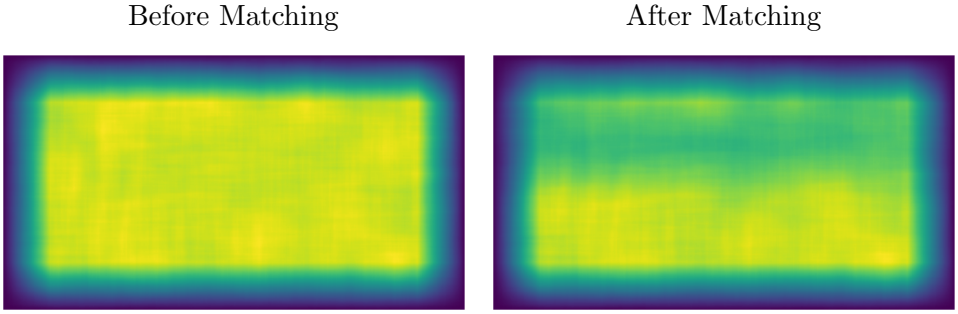


Figure 3.5: Heat map covered pixels by patches in the source domain where the y and x dimensions are the image shape. Each pixel indicates the sampling frequency after random sampling (left) and matching (right). Brighter pixels are sampled more often.

Richter et al. [RAHK22] weight matched patches such that image regions on the source domain are sampled evenly. Because better textural style-transfer of DA relevant classes can improve object detection performance,

weighting patches is enhanced with domain adaptive weighting in eq. 3.5.

$$w_{\text{car}} = \mu_w + \sigma_w \cdot k \tag{3.5}$$

Matches containing car patches are re-weighting by parameter k where μ_w is the mean and σ_w the standard deviation of the weights. Only car patches are re-weighted and not truck or bus patches, since segmentation pseudo-labels are unstable for rare classes. Additionally, the classes truck and bus make up only a small fraction of DA relevant pixels on the source domain, as shown in figure 3.4.

3.4 Metrics

In the following, metrics to calculate and evaluate both the matching are given. Additionally, metrics to evaluate style-transfer GANs are discussed.

3.4.1 Matching

To match patches, the cosine similarity S_C between two vectors x and y is calculated.

$$S_C(x, y) = \frac{x^T \cdot y}{\|x\| \cdot \|y\|} \tag{3.6}$$

The cosine distance D_C can be derived from the cosine similarity S_C, as given in eq. 3.7.

$$D_C(x, y) = 1 - S_C(x, y) \tag{3.7}$$

For efficiency, Richter et. [RAHK22] utilize the L_2-distance to calculate the cosine distance, which is derived in eq. 3.8 with x and y being L_2 normalized.

$$D_C(x, y) = 1 - x^T y = \frac{1}{2}\|x\|^2 + \frac{1}{2}\|y\|^2 - x^T y = \frac{1}{2}(x-y)^T(x-y) = \frac{1}{2}\|x - y\|^2 \tag{3.8}$$

To evaluate the matching, all matched patches are mapped to the feature spaces ϕ_{VGG} and ϕ_{SEM}. Then, the cosine similarity is calculated both on the mapping ϕ_{VGG} and ϕ_{SEM} and the average of all cosine similarities is calculated for all matches.

3.4.2 Perceptual Evaluation

Metrics to evaluate style-transfer GANs rely on the maximum mean discrepancy (MMD), which maps the vectors x and y into a Hilbert space $\mathcal{H}$ with a kernel $k(x, y)$ [GBR+12]. The squared MMD can be calculated with

an unbiased estimator[GBR+12], as given in eq. 3.9.

$$\begin{aligned}\mathrm{MMD}_u^2(X,Y) = &\frac{1}{m(m-1)}\sum_{i\neq j}^{m} k(x_i,x_j) \\ &+ \frac{1}{n(n-1)}\sum_{i\neq j}^{n} k(y_i,y_j) + \frac{2}{mn}\sum_{i=1}^{m}\sum_{j=1}^{n} k(x_i,y_j)\end{aligned} \tag{3.9}$$

The MMD compares two distributions with regards to their skewness as well as the mean and variance [BSAG18]. In the following, all discussed metrics calculate the squared MMD with a polynomial kernel $k(x,y) = (\frac{1}{d}x^T y + 1)^3$[BSAG18][RAHK22]. A commonly used metric is the Kernel Inception Distance (KID) [BSAG18], as stated in eq. 3.10.

$$\mathrm{KID}(X,Y) = \mathrm{MMD}_u^2(\phi_{\mathrm{INC}}(X), \phi_{\mathrm{INC}}(Y)) \tag{3.10}$$

The KID maps generated images X and real images Y with a InceptionV3 model[SVI+16] into feature space ϕ_{INC}.

Richter et al. [RAHK22] state, that the KID[HRU+17] focuses only on high semantic concepts. Thus, they propose the semantically aligned kernel VGG distance (sKVD) to focus on a perceptual evaluation of high- and low-level semantics by extracting feature vectors from different layers of the VGG16 network[SZ15]. The KID is evaluated on whole images, whereas Richter et. al [RAHK22] evaluate the sKVD on patches of approximately 12% the image size. Richter et. al [RAHK22] pre-compute semantic segmentation pseudo-labels with MSeg[LLS+20] for both datasets and sample patches from semantic segmentation masks, which are resized to 16x16. The resized patch is encoded into a vector and the nearest neighbor with most common entries found. Matches of patches belonging to the semantic segmentation masks are visualized in figure 3.6. Then, the MMD is calculated

Figure 3.6: Matches based on semantic segmentation masks to calculate the sKVD proposed by Richter et al. [RAHK22].

on VGG16[SZ15] features from matching patches, as stated in eq. 3.11.

$$\text{sKVD}_i(X, Y) = \text{MMD}_u^2(\phi^i_{\text{VGG}}(X), \phi^i_{\text{VGG}}(Y)) \tag{3.11}$$

The sKVD_i extracts VGG16[SZ15] feature vectors after the last ReLU of the i-th convolutional layer [RAHK22]. MSeg[LLS+20] is trained on real data and thus is substituted with ProDA[ZZZ+21] which is less stable with 57.5 compared to 76.3 mIoU on *Cityscapes*[COR+16]. Nevertheless, ProDA is trained without any labels on real data for domain adaption.

3.4.3 Semantic Consistency

The style-transfer GAN may align both domains perceptually well but change the textures, such that the content is also changed in addition to the style. Figure 3.7 shows a semantic consistent and inconsistent style-transfer GAN. The latter generates artifacts in the clouds and colorizes most of the image in a gray-tone, which is likely to affect object detection training negatively. Thus, semantic segmentation pseudo-labels L_S and L_F are extracted with ProDA[ZZZ+21] for all source and style-transferred images to evaluate semantic consistency by calculating the mIoU[EVGW+10], which can be calculated efficiently in eq.3.12, as proposed by [HTP+18].

$$\text{mIoU} = \frac{1}{N} \frac{\sum_i n_{ii}}{t_i + \sum_j n_{ji} - n_{ii}} \tag{3.12}$$

Let N be the number of classes and n_{ij} be the number of pixels predicted as class j belonging to class i [HTP+18]. The sum of all pixels given to a class is given as $t_i = \sum_j n_{ij}$. The mIoU calculates the right predictions over the union of false positive, true positive and false negative predicted pixels. As shown in figure 3.7, style-transfer leads to a lot false positives in the segmentation map which is already evaluated in the mIoU. Hence, the pixel accuracy is calculated in eq. 3.13 for semantic consistency, as proposed by Hoffman et al. [HTP+18].

$$\text{pixel acc.} = \frac{\sum_i n_{ii}}{\sum_i t_i} \tag{3.13}$$

Additionally, both the mIoU and pixel accuracy are calculated for the class car to evaluate style-transfer with regards to the downstream task.

(a) (b) (c)

Figure 3.7: Evaluation of semantic consistency with pseudo-labels from ProDA[ZZZ+21]. (a) shows an image from DiM that is style-transferred semantically consistent in (b) and inconsistent in (c).

Chapter 4

Implementation

4.1 Framework

The framework for sampling and matching patches is inspired and modified from the code[1] provided by Richter et al. [RAHK22]. An overview is given in figure 4.1.

First, crops are sampled with a fixed patch size. Different amounts of crops can be extracted for the source and target data set for each image, which are evaluated in the experiments. The sampling can be done randomly or also by sampling patches with domain adaption classes. For domain adaptive sampling, patches containing at least 10% of a domain adaption class (car, trucks and buses) are sampled. If possible, 50% domain adaptive patches are sampled while the remaining patches are sampled randomly.

Feature vectors are extracted for all sampled crops after conv_5 ReLU activation from the VGG16[SZ15] network, which is used commonly for perceptual tasks[YUC+19][ZIE+18][RAHK22]. The implementation of Richter

[1] `https://github.com/isl-org/PhotorealismEnhancement/tree/main/code`

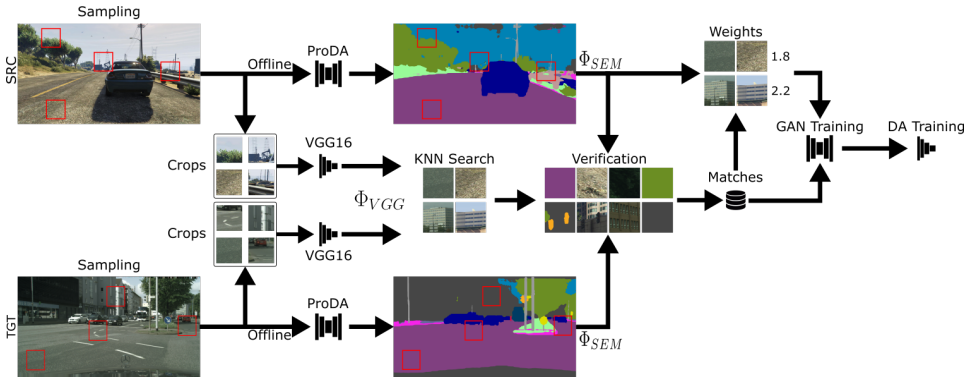


Figure 4.1: Patch sampling framework.

et al. [RAHK22] is used with no padding such that the receptive field of the feature extractor is 196 [2]. Generalized mean pooling [RTC18] can be applied for larger patch sizes than the receptive field or the image can be resized to the receptive field. If the given patch size is smaller than the receptive field, patches are sampled with the size of the receptive field and center cropped afterwards. As the patch size influences GAN training strongly, experiments are conducted with multiple patch sizes. Additionally, as mentioned in section 3.1 semantic *pdf* crops are extracted from precomputed ProDA[ZZZ+21] segmentation masks. Depending on the configuration, nearest neighbor search and verification is performed on perceptual or semantic *pdf* feature vectors. A matching based on ϕ_{SEM} lacks diversity because patches containing only one class are always matched to the same k patches containing the same class. Hence, k nearest neighbor search is performed with a big k and the one with the highest cosine similarity based on ϕ_{VGG}(as given in eq. 3.1) is chosen.

KNN search is performed on feature vectors with FAISS [JDJ19], which provides fast batch search utilizing the GPU. For every crop of the source domain, k nearest neighbors are found based on the cosine similarity. Analog to Richter et al. [RAHK22], the L2 distance after normalization (see eq. 3.7) is used for efficiency. FAISS search time scales approximately proportional with sampling size, such that the largest configuration finds the nearest neighbor for 1,300,000 patches in the source domain searching 900,000 patches in the target domain in approximately 3 hours on a regular setup[3]. Alternatively, FAISS provides approximate nearest neighbor search to reduce runtime if larger configurations are evaluated. Using the nearest neighbors, the verification similarities are calculated on sampled semantic *pdf* vectors ϕ_{SEM} such that neighbors can be filtered regarding search and verification threshold. Matched image pairs are stored together with their source images and locations for GAN training. Matches are weighted as proposed by Richter et al. [RAHK22]. Afterwards, matches which contains at least 10% car pixels in the source patch, can be re-weighted according to section 3.3. The matched crops and matching weights are then used to train the style-transfer GAN, which style-transfers source images for object detection training.

4.2 Training

Vehicle detection is learned on the Sim10K split from the synthetic dataset *Driving in the Matrix*(DiM)[JRBM+17] and evaluated on the real dataset *Cityscapes*(CS)[COR+16]. Sim10K contains night images, which cause the

[2] `https://github.com/Fangyh09/pytorch-receptive-field/blob/master/torch_receptive_field/receptive_field.py`

[3] Workstation with two NVIDIA Titan X and an Intel i7-5930K processor

GAN to degenerate because CS only has day images. Hence, a filtered dataset of Sim10K is used with day images only. For style-transfer, a CycleGAN[ZPIE17] is trained with DiM as source dataset and CS as target dataset. Parameters are optimized with ADAM with the learning rate set to 0.0002, β_1 set to 0.5 and β_2 set to 0.99. The model is trained for 200 epochs, in which the first 100 epochs the learning rate is kept constant. Afterwards, the learning rate is updated with a Lambda scheduler, which decreases the learning rate the last 100 epochs linearly to 0. No identity loss is applied, λ_A and λ_B are set as losses to 10. The matches are pre-computed and assembled to a batch size of 2 for training. Weights for matches are divided by the cumulative sum of all weights and a match chosen with a random number in the range from 0 to 1, as proposed by Richter et al. [4].

The Faster-RCNN[RHGS15] model is trained for object detection with a VGG16 backbone[SZ15]. Images are style-transferred and then the object detector trained with a batch size 2 for 20 epochs. ADAM is used for parameter optimization with learning rate set to 0.0001, β_1 set to 0.9 and β_2 set to 0.999. The learning rate is updated by a step learning rate with step size of 10. Loss is weighted with λ_{RPN} set to 10.0 and λ_{Box} set to 1.0. For evaluation, images are resized to 512 and the mAP calculated as proposed for the *MS COCO* dataset [LMB+14], e.g. predicted boxes are correct if the intersection over union with ground-truth boxes is over 0.5. Since detection performance varies depending on the seed, three different seeds are evaluated for every trained style-transfer GAN and mean results are reported.

4.3 Configurations

Implemented configurations for patch sampling are listed in table 4.1. The column *Weighting* indicates the weighting method discussed in chapter 3.3. If domain adaptive weighting is applied, the column *Reweighting Value* shows the weight k for eq. 3.5. *KNN* indicates, how many nearest neighbors are searched for, such that the first nearest neighbor above the *verification threshold* is chosen. *Matches Per Patch* determines, how many matched and verified target patches are kept for every source patch. The column *Search* sets the feature space for KNN-search, whereas *Verification* determines the feature space for verification. If search is performed on ϕ_{VGG}, *Sem Threshold* sets t_{SEM} according to eq. 3.3. *Sampling* sets the sampling algorithm for sampling patches. *#samples src* and *#samples tgt* sets the sampling sizes for the source and and target domain, e.g. the amount of crops per image. The column patch size *Patch Size* sets the size of sampled patches, whereas *Size Policy* determines how patches unequal the receptive field are handled.

[4] `https://github.com/isl-org/PhotorealismEnhancement/blob/main/code/epe/matching/paired.py`

Configuration	Reweighting Value	KNN	Matches Per Patch	Search	Sem Threshold	Verification	Sampling	#samples src	#samples tgt	Patch Size	Weighting	Size Policy
real_random*	-	-	-	-	-	-	-	-	-	196	-	-
epe	-	1	1	VGG16	-	-	random	50	50	196	EPE	-
epe_p128	-	1	1	VGG16	-	-	random	50	50	**128**	EPE	**Center Crop**
epe_rp256	-	1	1	VGG16	-	-	random	50	50	**256**	EPE	**Resize To RF**
epe_p256	-	1	1	VGG16	-	-	random	50	50	**256**	EPE	**GeM Pool**
epe_rp	-	1	1	VGG16	-	-	random	50	50	**392**	EPE	**Resize To RF**
epe_p512	-	1	1	VGG16	-	-	random	50	50	**512**	EPE	**GeM Pool**
epe_s50t100	-	1	1	VGG16	-	-	random	50	**100**	196	EPE	-
epe_s50t250	-	1	1	VGG16	-	-	random	50	**250**	196	EPE	-
epe_s50t300	-	1	1	VGG16	-	-	random	50	**300**	196	EPE	-
epe_s50t300_knn4	-	**4**	**4**	VGG16	-	-	random	50	**300**	196	EPE	-
epe_s50t300_daw9	**9**	1	1	VGG16	-	-	random	50	**300**	196	EPE	-
epe_s50t350	-	1	1	VGG16	-	-	random	50	**350**	196	EPE	-
epe_s200t50	-	1	1	VGG16	-	-	random	**200**	50	196	EPE	-
epe_s200t300	-	1	1	VGG16	-	-	random	**200**	**300**	196	EPE	-
epe_s200t300_sem_t001	-	1	1	VGG16	**0.01**	**SEM PDF**	random	**200**	**300**	196	EPE	-
epe_s200t300_daw9	**9**	1	1	VGG16	-	**SEM PDF**	random	**200**	**300**	196	**DA**	-
epe_daw3	**3**	1	1	VGG16	-	-	random	50	50	196	**DA**	-
epe_daw6	**6**	1	1	VGG16	-	-	random	50	50	196	**DA**	-
epe_daw9	**9**	1	1	VGG16	-	-	random	50	50	196	**DA**	-
epe_das	-	1	1	VGG16	-	-	**DAS**	50	50	196	EPE	-
sem	-	1	1	**SEM PDF**	-	-	random	50	50	196	EPE	-
epe_sem_t001	-	25	1	VGG16	**0.01**	**SEM PDF**	random	50	50	196	EPE	-
epe_sem	-	**100**	1	VGG16	**0.5**	**SEM PDF**	random	50	50	196	EPE	-
sem_epe	-	**80**	1	**SEM PDF**	-	**VGG16**	random	50	50	196	EPE	-

Table 4.1: Patch sampling configurations for GAN training. For ablation studies, only one parameter is changed if possible to compare different configurations. Parameters are marked bold that are changed in comparison to the *epe* configuration. **real_random* is the standard training configuration without a patch sampling pipeline. For each iteration during GAN training, an image and then a patch is sampled randomly for the source and target domain.

Chapter 5

Experiments

Experiments for CycleGAN training are evaluated in this chapter with configurations described in section 4.3. Cycle-GAN[ZPIE17] training is unstable and hyperparameters are hard to fine-tune because GAN training takes one week. Hence, results fluctuate heavily and multiple GAN trainings are evaluated in section 5.4 for the main mAP baselines.

5.1 Matching

Figure 5.1 shows an example *Driving in the Matrix*(DiM)[JRBM+17] and *Cityscapes*[COR+16] image to visualize the different domain biases. In *Cityscapes*, streets are textured smoother and grayer, trees are colored in a darker green or the sky is often overexposed. DiM on the other hand contains images from both the countryside and city with a subtropical climate in comparison to the German temperate climate in *Cityscapes*.

(a) Driving in the Matrix **(b)** Cityscapes

Figure 5.1: Example images from *Driving in the Matrix*(DiM)[JRBM+17] and *Cityscapes*[COR+16]. Streets in *DiM*(a) are textured coarser than streets from *Cityscapes*. Additionally, colors in *Cityscapes*(a) are more saturated which can be seen in the vegetation and streets.

Hence, style-transfer aims to align the source domain with the target domain, as seen in figure 5.2. Mostly, style-transferred images such as (b)

and (c) are generated content consistent and it looks like a gray filter is applied by darkening bright colors such as the brown of the road or the blue of the sky. However, style-transfer GANs can hallucinate artifacts such as generated specular light on the road in (b).

(a) Driving in the Matrix **(b)** EPE **(c)** Real Random

Figure 5.2: Example *DiM*[JRBM$^+$17] image (a) style-transferred by a CycleGAN[ZPIE17] which is trained on matches from the configurations *epe* (b) and *real_random* (c). The configurations (b) and (c) learn to darken bright colors such as brown or blue.

An additional verification of matches with a semantic *pdf* threshold causes the GAN to colorize streets greener which is shown in figure 5.3. While the magnitude varies strongly depending on the scene, the configurations *epe_sem*, *epe_sem_t001* and *epe_s200t300_t001* with a semantic *pdf* verification threshold learned this effect.

(a) Driving in the Matrix **(b)** EPE **(c)** EPE SEM

Figure 5.3: Example *DiM*[JRBM$^+$17] image (a) style-transferred by a CycleGAN[ZPIE17] which is trained on matches from the *epe* (b) and *epe_sem* (c) configuration. The configuration *epe* (b) without semantic verification stylizes streets less green than the configuration *epe_sem* with semantic verification (c).

Matching patches in ϕ_{SEM} causes the generator to change less, whereas an alignment with ϕ_{VGG} causes the generator to learn a darkener style-transfer. Figure 5.4 shows the darkening, as the blue sky is darkened by the configuration *epe* whereas the configuration *sem_epe* even adds bright gray colors to the sky. Both configurations (b) and (c) change yellow colors by whitening the street markers or making the soil more gray.

Chapter 4.1 discusses that ϕ_{SEM} feature space is under-determined such that the search space to match patches is too small. Therefore, the configuration *sem* lacks diversity and the generator generates wave artifacts which can be seen in figure 5.5.

Table 5.1 shows average ϕ_{SEM} and ϕ_{VGG} cosine similarities of matches

(a) Driving in the Matrix **(b)** EPE **(c)** SEM EPE

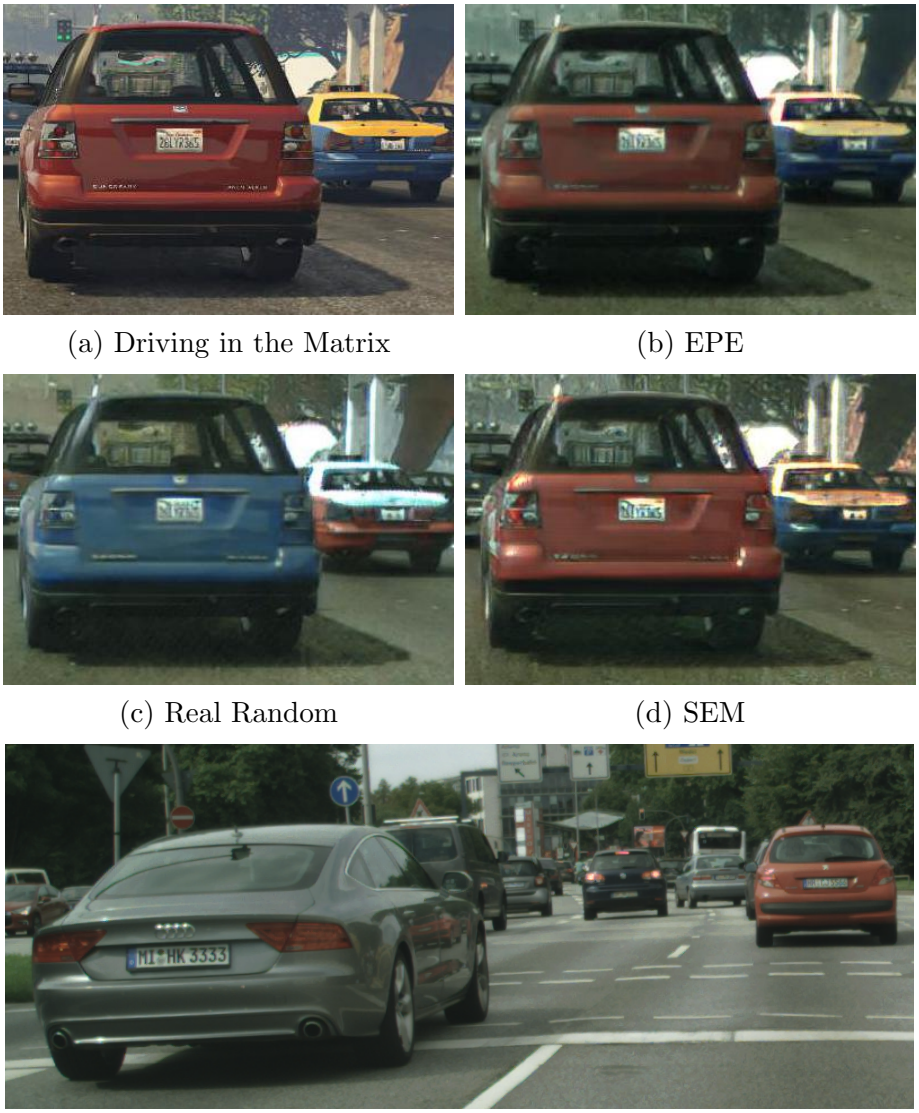
Figure 5.4: Example *DiM*[JRBM+17] image (a) style-transferred by a CycleGAN[ZPIE17] which is trained on matches from the configurations *epe*(b) and *sem_epe* (c). The configuration *epe* with a ϕ_{VGG} feature search generates darker images than the configuration *sem_epe* with a ϕ_{SEM} feature search.

which are used for GAN training. Sampling patches random results in a low cosine similarity on VGG16[SZ15] and semantic *pdf* feature vectors. Matching patches with VGG16[SZ15] feature vectors improves the average ϕ_{VGG} and ϕ_{SEM} cosine similarity significantly in comparison to random sampling. Performing a search on semantic *pdf* vectors increases VGG16 only slightly, while reaching an average ϕ_{SEM} cosine similarity of 0.99. The configurations *epe_sem* and *epe_sem_t001* apply a semantic *pdf* verification threshold and thus improve the average ϕ_{SEM} cosine similarity while preserving average ϕ_{VGG} cosine similarity.

Method	random	epe	epe_sem	epe_sem_t001	sem	sem_epe
Average VGG16 cos-sim	0.1010	0.7444	0.7400	0.7426	0.1917	0.5243
Average Sem. *pdf* cos-sim	0.3263	0.6979	0.8998	0.7824	0.9921	0.9851
Number of Matches	329550	317502	302578	311306	329505	328443

Table 5.1: Average VGG16[SZ15] and semantic *pdf* cosine similarity of matching configurations based on different search and verification policies. Matching based on ϕ_{VGG} features aligns both domains in ϕ_{VGG} and ϕ_{SEM} feature space, while matching based on ϕ_{SEM} only slightly increases average VGG16 cosine similarity. Additionally, the number of matches show that configurations *epe_sem* and *epe_sem* filter out approximately 9% patches in comparison to the configuration *random* without matching.

Table 5.2 shows that matching in ϕ_{VGG} improves semantic consistency shown in the better car pixel accuracy and pixel accuracy in comparison to the configuration *real_random*. The configuration *real_random* learned a red-blue inversion and thus holds the lowest scores in semantic consistency metrics, while it achieves some of the best sKVD values. The configuration *epe* beats the random baseline by approximately 0.9 mAP. Matching only on ϕ_{SEM} features (configuration *sem*) reaches higher semantic consistency than the random baseline but performs worse on the sKVD_1 and sKVD_2, while holding worse mAP than the random baseline. The perceptual enhanced configuration *sem_epe* reaches in most semantic consistency met-



(a) Driving in the Matrix (b) EPE

(c) Real Random (d) SEM

(e) Cityscapes

Figure 5.5: Example *DiM*[JRBM+17] image (a) style-transferred by a CycleGAN[ZPIE17] trained on the configurations *epe*(b), *real_random*(c) and *sem*(d). All configurations brighten the taxi, remove fine-grained information by blurring car signs and dampening reflections on the red car. Image (c) shows blue-red inversion from the configuration *random*. The perceptual under-determined configuration *sem* (d) generates less detailed image with wave artifacts in the red car and taxi. A reference *Cityscapes*(e) image shows cars which are less saturated in comparison to *DiM* cars.

rics the highest scores and better mAP than the configuration *sem*, while the mAP, car pixel accuracy and $sKVD_2$ are worse than the *epe* baseline.

Refining the configuration *epe* with semantic *pdf* verification (*epe_sem* and *epe_sem_t001*) worsens semantic consistency values and thus decreases mAP.

Configuration	$sKVD_5$	$sKVD_4$	$sKVD_3$	$sKVD_2$	$sKVD_1$	mIoU	pixel accuracy	Car mIoU	Car pixel accuracy	mAP
real_random	**0.0026**	0.0048	0.0298	**0.0058**	**0.0004**	0.3528	0.4953	0.6170	0.7591	47.6600
epe	0.0043	0.0068	0.0373	0.0268	0.0016	0.4502	0.5767	0.6824	**0.8136**	**48.5633**
epe_sem	0.0030	**0.0040**	**0.0231**	0.0147	0.0009	0.4013	0.5353	0.6409	0.7600	47.2400
epe_sem_t001	0.0039	0.0066	0.0687	0.0457	0.0025	0.4030	0.5375	0.6484	0.7633	47.1533
sem	0.0033	0.0049	0.0295	0.0340	0.0010	0.4519	0.5758	0.6771	0.7602	46.3933
sem_epe	0.0034	0.0056	0.0353	0.0309	0.0012	**0.4761**	**0.5879**	**0.6904**	0.7642	47.8500

Table 5.2: Semantic and textural evaluation of matching configurations based on different search and verification policies. The configuration *epe* based on VGG16[SZ15] features only holds the highest mAP and beats the previous random baseline. However, applying additional semantic verification (*epe_sem* and *epe_sem_t001*) or searching on ϕ_{SEM} (*sem* and *sem_epe*) does not improve object detection results.

5.2 Sampling

5.2.1 Domain Adaptive Sampling

Figure 5.6 shows that DA sampling approximately triples the pixel percentage of sampled DA classes while falling short on common classes such sky and road.

The change of dataset bias causes the GAN to degenerate by learning a red-blue inversion. Figure 5.7 shows an example image of the degeneration that darkens the image strongly. Hence, all semantic consistency metrics drop in comparison to the configuration *epe* which results in a lower mAP, as seen in table 5.3.

Configuration	$sKVD_5$	$sKVD_4$	$sKVD_3$	$sKVD_2$	$sKVD_1$	mIoU	pixel accuracy	Car mIoU	Car pixel accuracy	mAP
epe	**0.0043**	**0.0068**	0.0373	0.0268	**0.0016**	**0.4502**	**0.5767**	**0.6824**	**0.8136**	**48.5633**
epe_das	0.0070	0.0168	**0.1569**	**0.0264**	0.0027	0.2956	0.4347	0.5364	0.6979	47.3800

Table 5.3: Semantic and textural evaluation of configurations with different sampling algorithms. The configuration *epe_das* with domain adaptive sampling is semantically inconsistent which lowers mAP in comparison to the *epe* baseline with random sampling.

5.2.2 Patch Size

The patch size determines whether more local or global information is shown to the GAN and has a big impact on GAN training. Larger patch sizes can be resized to the receptive field or GeM[RTC18] pooled. Figure 5.8 illustrates, that resizing larger patch sizes hardly effects the cosine similarity density distribution. GeM pooling on the other hand generates a normal distribution for larger patch sizes (13x13 pooling for patch size 392 and 4x4

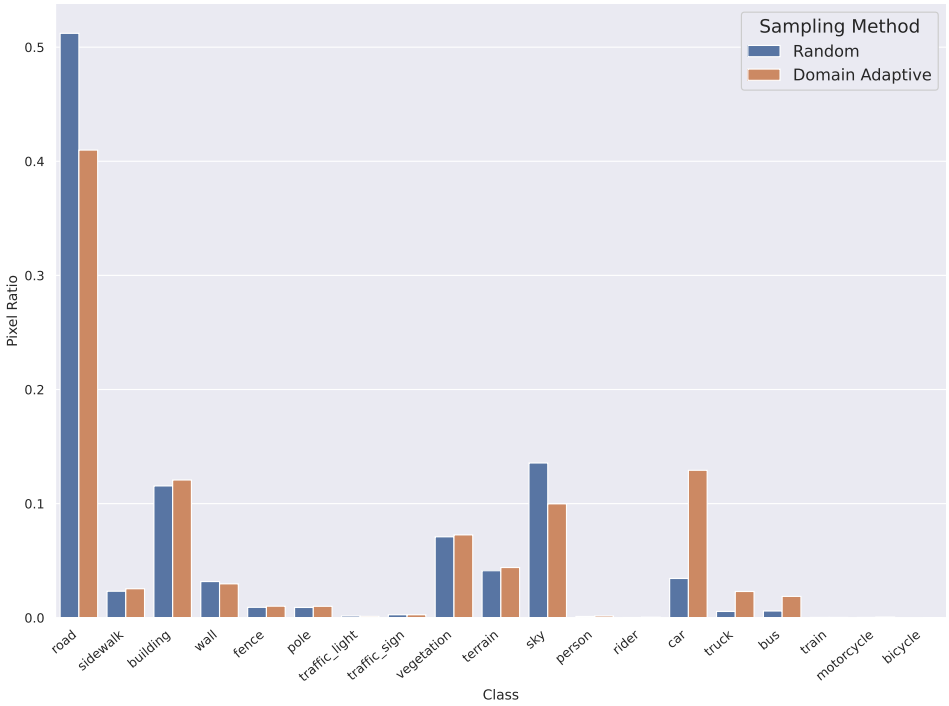


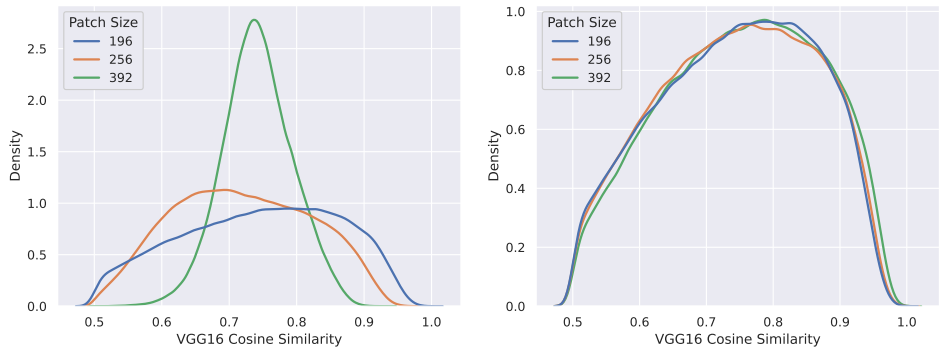
Figure 5.6: Class distribution of patches sampled randomly or with a domain adaptive (car, truck and bus pixels) sampling algorithm from *Driving in the Matrix*[JRBM+17](DiM). Domain adaptive sampling approximately triples the pixel ratio of domain adaptive classes while classes such as road and sky are sampled less in comparison to random sampling.

(a) Driving in the Matrix **(b)** EPE DAS

Figure 5.7: Example *DiM*[JRBM+17] image (a) style-transferred by a CycleGAN[ZPIE17] trained on the configurations *epe_das*(b). The domain adaptive sampling configuration (b) learns a blue-red inversion and darkens the image such that a lot of information is lost.

pooling for patch size 256). Hence, GeM pooling for large patches lead to a worse matching due to densely distributed feature vectors and should only be applied on smaller patch sizes such as 256.

Larger patch sizes cause the GAN to change more content aside from



(a) GeM Pooling **(b)** Resizing

Figure 5.8: VGG16[SZ15] cosine similarity density distribution of matches based on feature vectors extracted by GeM pooling[RTC18] (a) and by resizing patches to the receptive field(b). GeM pooling on larger patch sizes results in a dense Gaussian distribution of feature vectors, while the density distribution is stable for resizing.

the style and thus leads to artifacts. Figure 5.9 shows an example for the configuration *epe_p512* which removes parts of the left car and the person in the right car.

(a) Driving in the Matrix **(b)** Patch Size 512

Figure 5.9: Example *DiM*[JRBM+17] patch (a) style-transferred by a CycleGAN[ZPIE17] which is trained on matches from the configurations *epe_p512*. Large patch sizes cause the GAN to generate more artifacts such as hallucinated vegetation at the bottom left of (b) or even changing shapes of the cars.

The amount of artifacts is dependent on the scene and can change a lot even though the scene only changes a little. Thus, more global content causes less mismatches to exist with a cosine similarity of 0 which is visualized in figure 5.10, where the local density extrema vanishes at 0 for the patch size 512.

Experiments with smaller patch sizes show that configurations with a patch size equal or less 196 do not create layout artifacts and only change style. Figure 5.11 shows a comparison of trained configurations with different patch sizes at which the configurations for the sizes 128 and 512 learned a darkening as seen in the white car. The patch sizes 256 on the other hand

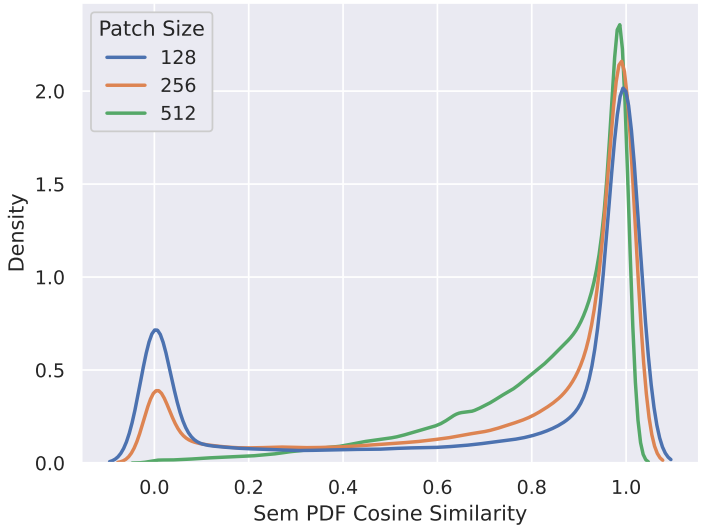


Figure 5.10: Semantic *pdf* cosine similarity density distribution of matches based on the patch size. Larger patch sizes lead to less mismatches as seen in the density peaks at cosine similarity 0.

adds specular reflections to cars, as seen in the red car.

A property of style-transfer GAN, where the scene only changes a little but far more artifacts are generated is enhanced with larger patch sizes, as figure 5.12 shows. In this case more artifacts in the clouds and the sky gets colored more gray-green on the right picture.

Evaluation results in table 5.4 and figure 5.12 show that the GAN trained on configuration *epe_p128* learns to darken images. This results in a lower mAP, car pixel accuracy and mIoU in comparison to the configuration *epe*. Increasing the patch size to 256, 392 and 512 causes major decrease in semantic consistency metrics which may be due to hallucinations. However, the configurations *epe_p256* and *epe_rp256* achieve both slightly worse mAP in comparison to the *epe* baseline. The configuration *epe_p256* with GeM[RTC18] pooling style-transfers images semantically more consistent and perceptually better than the configuration *epe_rp256* without pooling. Despite that, the configuration *epe_rp256* holds slightly better mAP and takes less compute-power which makes GeM pooling obsolete.

5.2.3 Sampling Size

Increasing the sampling size on the target domain causes a logarithmic growth in average VGG16[SZ15] cosine similarity, as shown in table 5.5. Additionally, the feature extraction and KNN-search time increases, such that a balance between performance and compute time has to be found. As the source domain dataset *Driving in the Matrix*[JRBM+17] has twice as

(a) Patch Size 128

(b) Patch Size 196

(c) Patch Size 256

(d) Patch Size 512

(e) Driving in the Matrix

Figure 5.11: Example *DiM*[JRBM+17] image (e) style-transferred by a CycleGAN[ZPIE17] trained on the configurations with different patch sizes. The configuration *epe_p128*(a) darkens the image, whereas the configurations *epe*(b) and *epe_p256*(c) generate brighter images and even add light to the bright white car. The configuration *epe_p512*(d) learned a blue-red inversion.

Configuration	$sKVD_5$	$sKVD_4$	$sKVD_3$	$sKVD_2$	$sKVD_1$	mIoU	pixel accuracy	Car mIoU	Car pixel accuracy	mAP
epe_p128	0.0031	0.0102	0.1040	0.0134	0.0013	0.3823	0.5148	0.6177	0.7384	47.2867
epe	0.0043	0.0068	0.0373	0.0268	0.0016	**0.4502**	**0.5767**	**0.6824**	**0.8136**	**48.5633**
epe_rp256	0.0039	0.0121	0.1095	0.0169	0.0016	0.3482	0.4828	0.5855	0.7207	48.3467
epe_p256	**0.0022**	**0.0032**	0.0193	0.0046	0.0007	0.3750	0.5211	0.6446	0.7573	48.3233
epe_rp392	0.0025	0.0035	**0.0184**	**0.0036**	**0.0006**	0.3004	0.4663	0.5517	0.7541	47.0000
epe_p512	0.0045	0.0112	0.0856	0.0151	0.0007	0.2826	0.4519	0.5315	0.7298	46.1333

Table 5.4: Semantic and textural evaluation of configurations with different patch sizes. Patch sizes larger than 256 and smaller than 196 degenerate, resulting in lower mAP. Resizing and pooling achieve almost the same mAP scores which makes resizing more feasible due to less computation requirements.

much images as *Cityscapes*[COR+16], increasing the sampling size on the source domain causes 3-times higher computation time. Sampling more on the source domain does not influence the matching but increases the number

Figure 5.12: Two sequential images from *DiM*[JRBM+17] which are style-transferred with a CycleGAN[ZPIE17] trained on the configuration *epe_p512*. Style-transfer can change drastically if the scene changes slightly, as both color brightness and hallucinated artifacts differ on the style-transferred images.

of distinct matches for GAN training.

Method	Compute Time (min)	Average VGG cos-sim	mAP	mAP per Compute Time (min)
epe	37.2267	0.7444	48.5633	1.3045
epe_s50t100	52.5494	0.7553	46.6600	0.8879
epe_s50t250	99.8516	0.7700	47.2700	0.4734
epe_s50t300	115.3271	0.7729	49.0700	0.4255
epe_s50t350	130.7996	0.7752	47.3500	0.3620
epe_s50t300_knn4	115.3386	0.7556	49.2000	0.4266
epe_s200t50	111.6789	0.7445	48.9267	0.4381
epe_s200t300	312.5121	0.7731	47.1267	0.1508

Table 5.5: Compute time in relation to the VGG16[SZ15] and object detection performance. Configurations that sample more (*epe_s50t300* and *epe_s50t300_knn4*) hold the highest mAP, while the mAP per compute time efficiency sinks. However, results are unstable such and further experiments have to be evaluated.

Table 5.6 shows the instability of CycleGAN[ZPIE17] training. While only the sampling rate on the source and target domain are increased, all metrics vary a lot. Sampling more on the target domain improves ϕ_{VGG} feature alignment of matches which should increase perceptual style-transfer. However, the configuration *epe_s50t250* matches patches slightly worse than the configuration *epe_s50t300* (see figure 5.5) and achieves significantly less mAP. Similar, sampling more on the source domain is unstable, as the configuration *epe_s200t300* reaches less mAP than the *epe* baseline. Thus, evaluation of the sampling size is difficult and more experiments have to be performed to confirm mAP performance.

Configuration	$sKVD_5$	$sKVD_4$	$sKVD_3$	$sKVD_2$	$sKVD_1$	mIoU	pixel accuracy	Car mIoU	Car pixel accuracy	mAP
epe	0.0043	0.0068	0.0373	0.0268	0.0016	0.4502	0.5767	0.6824	**0.8136**	48.5633
epe_s50t100	0.0027	**0.0035**	**0.0186**	0.0142	**0.0002**	0.3972	0.5366	0.6353	0.7693	46.6600
epe_s50t250	0.0048	0.0064	0.0462	0.0508	0.0027	0.4502	**0.5858**	**0.6953**	0.7886	47.2700
epe_s50t300	0.0028	0.0061	0.0320	0.0172	0.0009	**0.4532**	0.5823	0.6845	0.7996	49.0700
epe_s50t350	0.0050	0.0140	0.1232	0.0202	0.0020	0.3668	0.5056	0.6113	0.7423	47.3500
epe_s50t300_knn4	0.0024	0.0064	0.0555	0.0083	0.0007	0.3871	0.5224	0.6477	0.7744	**49.2000**
epe_s200t50	**0.0019**	0.0036	0.0208	**0.0080**	0.0004	0.4179	0.5527	0.6606	0.7660	48.9267
epe_s200t300	0.0041	0.0061	0.0407	0.0349	0.0016	0.4491	0.5782	0.6865	0.7910	47.1267

Table 5.6: Semantic and textural evaluation of configurations with different sampling sizes on the source and target domain. Sampling more on both domains results in a higher mAP, as seen for trainings based on the configurations *epe_s200t50* and *epe_s50t300*. As mAP and style-transfer metrics vary a lot, more experiments have to be conducted for confirmation.

Experiments show that some configurations generate artifacts around car tires and cars itself, as seen in figure 5.13. This occurs rarely and does not lead to a decrease of mAP, as for example the configuration *epe_s50t300* learned and still improves detection results (see table 5.6).

(a) Driving in the Matrix **(b)** EPE **(c)** Artifacts

Figure 5.13: Example *DiM*[JRBM+17] image (a) style-transferred by a CycleGAN[ZPIE17] trained on the configurations *epe*(b) and *epe_s50t300*(c). Style-transfer GANs rarely generate wave patterns and noise (c) on car surfaces and tires. Mostly, style-transfer is stable as seen in (b).

5.3 Weighting

Rarely sampled regions are up weighted to improve style-transfer. Figure 5.14 shows the class wise pixel distribution for matches with no weights, EPE [RAHK22] weighting and domain adaptive weighting, in this case with $k = 9.0$. EPE weighting causes a down-weighting of car and street pixels, whereas heavy domain adaptive weighting causes a significant increase of car, truck and bus pixels by approximately 3 times. On the other side, road patches are down-weighted a lot more in comparison to EPE weighting.

Domain adaptive weighting causes the GAN to generate artificial lights at the car blinker which is illustrated in figure 5.15. Additionally, little light dots appear as artifacts at the left of the sign in figure 5.15(c). The EPE weighting on the other side adds no specular reflections to the car surface.

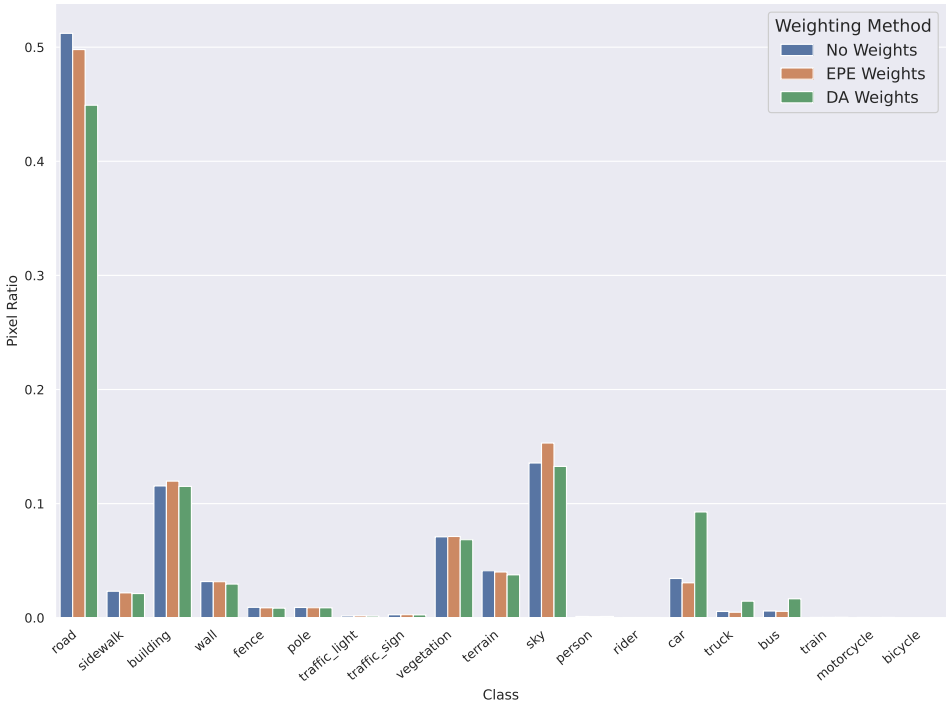
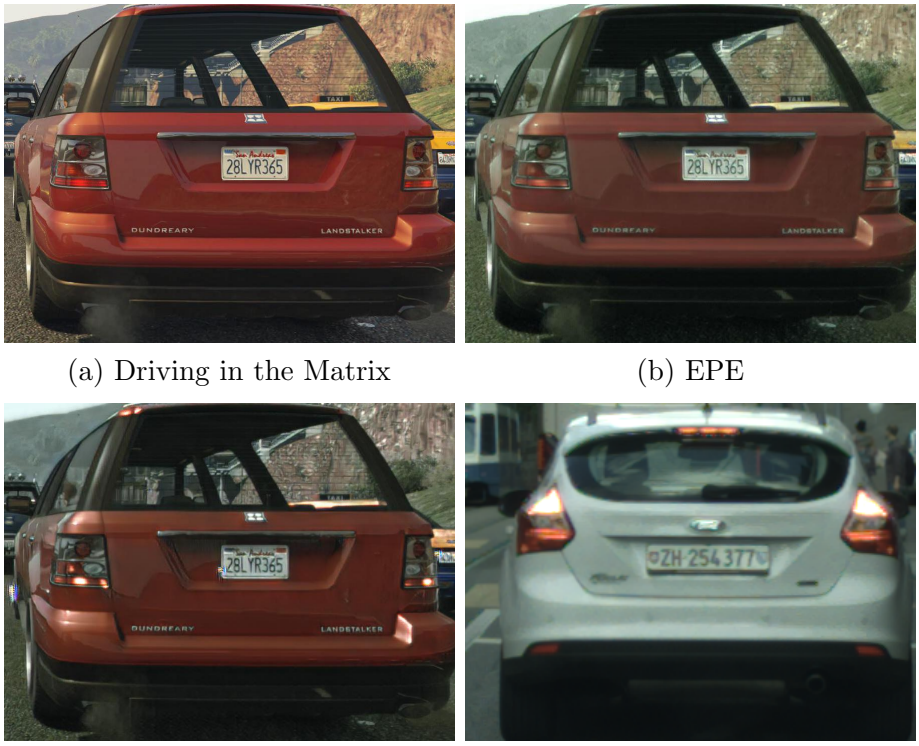


Figure 5.14: Class distribution of matches from the configuration *epe* with no weighting, weighting proposed by Richter et al. [RAHK22] and domain adaptive weighting with $k = 9.0$ according to section 3.3. Classes relevant for domain adaption are up-weighted by domain adaptive weighting while common classes are down-weighted in comparison to weighting proposed by Richter et al. [RAHK22].

The configurations *epe_daw3*, *epe_daw6* and *epe_daw9* all hold lower values in semantic consistency metrics, as table 5.7 shows. However, the configuration *epe_daw9* achieves better values in the sKVD_1 and sKVD_2 showing a better perceptual style-transfer. The best configuration *epe* slightly beats *epe_daw9* in domain adaption performance, whereas *epe_daw9* beats the average *epe* mAP. Minor domain adaptive weighting done by the configurations *epe_daw3* and *epe_daw6* does not improve mAP. Hence, more experiments have to be evaluated whether large domain adaptive weighting improves mAP.

Configuration	sKVD_5	sKVD_4	sKVD_3	sKVD_2	sKVD_1	mIoU	pixel accuracy	Car mIoU	Car pixel accuracy	mAP
epe	0.0043	0.0068	0.0373	0.0268	0.0016	**0.4502**	**0.5767**	**0.6824**	**0.8136**	**48.5633**
epe_daw3	0.0024	0.0059	0.0448	0.0101	0.0003	0.4392	0.5621	0.6806	0.7715	47.8467
epe_daw6	0.0041	0.0055	**0.0282**	0.0270	0.0012	0.3835	0.5126	0.6306	0.7394	47.7200
epe_daw9	**0.0022**	**0.0048**	0.0305	**0.0059**	**0.0002**	0.4263	0.5574	0.6478	0.7717	48.5000

Table 5.7: Semantic and textural evaluation of configurations with different weighting strategies. The configuration *epe_daw9* reaches almost the same mAP as the best configuration *epe*, whereas configurations with minor weightings perform worse than the *epe* baseline.



(a) Driving in the Matrix (b) EPE

(c) Domain Adaptive Weighting (d) Cityscapes

Figure 5.15: Example *DiM*[JRBM+17] image (a) style-transferred by a CycleGAN[ZPIE17] trained on the configurations *epe*(b) and *epe_daw9*(c). The configuration *epe_daw9* (c) with domain adaptive weighting puts more specular lights on car surfaces and rarely hallucinates lights into the blinker. The configuration without domain adaptive weighting (b) adds less specular hallucinations to the car. Cars in the real dataset *Cityscapes*[COR+16](d) have the blinker lights more often on in comparison to (a).

5.4 Overview

In addition to previous experiments, configurations *epe_sem_s200t300_t001* and *epe_s200t300_daw9* are deployed for training to combine semantic verification and domain adaptive weighting with larger sampling configurations. Table 5.8 shows an overview of all deployed patch sampling configurations for style-transfer and domain adaption.

The configuration *epe_s50t300_knn4* achieves the best mAP performance with 49.2 mAP. Good performing configurations such as *epe_s50t300_knn4* *epe_s200t50* show low $sKVD_1$ and $sKVD_2$ values, while the better performing configuration *epe_s50t300_knn4* holds a higher car pixel accuracy. The second-best configuration *epe_s50t300* shows higher $sKVD_1$ and $sKVD_2$ values than the best performing configuration, while achieving better car pixel

accuracy. These results show that higher car pixel accuracy is better while $sKVD_1$ and $sKVD_2$ values are also important to achieve a high mAP. Despite this, the mIoU and pixel accuracy may not drop too low, as seen for configuration *epe_rp392*. The configuration *epe_rp392* achieves good $sKVD_1$, $sKVD_2$ and car pixel accuracy values, while the mIoU at approximately 30 indicate artifacts in style-transfer. Hence, low-semantic perceptual alignment shown by the $sKVD_1$ and $sKVD_2$ indicate a correlation with better mAP. In addition, high semantic-alignment through car pixel accuracy shows improving results, while the mIoU should not drop too low due to resulting content inconsistency.

Configuration	$sKVD_5$	$sKVD_4$	$sKVD_3$	$sKVD_2$	$sKVD_1$	mIoU	pixel accuracy	Car mIoU	Car pixel accuracy	mAP
GTA	0.0306	0.0606	0.4312	0.2746	0.0353	1.0000	1.0000	1.0000	1.0000	33.3167
real_random	0.0026	0.0048	0.0298	0.0058	0.0004	0.3528	0.4953	0.6170	0.7591	47.6600 (46.7100*)
epe	0.0043	0.0068	0.0373	0.0268	0.0016	0.4502	0.5767	0.6824	**0.8136**	48.5633 (48.0522**)
epe_sem	0.0030	0.0040	0.0231	0.0147	0.0009	0.4013	0.5353	0.6409	0.7600	47.2400
epe_sem_t001	0.0039	0.0066	0.0687	0.0457	0.0025	0.4030	0.5375	0.6484	0.7633	47.1533
sem	0.0033	0.0049	0.0295	0.0340	0.0010	0.4519	0.5758	0.6771	0.7602	46.3933
sem_epe	0.0034	0.0056	0.0353	0.0309	0.0012	**0.4761**	**0.5879**	**0.6904**	0.7642	47.8500
epe_daw3	0.0024	0.0059	0.0448	0.0101	0.0003	0.4392	0.5621	0.6806	0.7715	47.8467
epe_daw6	0.0041	0.0055	0.0282	0.0270	0.0012	0.3835	0.5126	0.6306	0.7394	47.7200
epe_daw9	0.0022	0.0048	0.0305	0.0059	**0.0002**	0.4263	0.5574	0.6478	0.7717	48.5000
epe_das	0.0070	0.0168	0.1569	0.0264	0.0027	0.2956	0.4347	0.5364	0.6979	47.3800
epe_p128	0.0031	0.0102	0.1040	0.0134	0.0013	0.3823	0.5148	0.6177	0.7384	47.2867
epe_rp256	0.0039	0.0121	0.1095	0.0169	0.0016	0.3482	0.4828	0.5855	0.7207	48.3467
epe_p256	0.0022	**0.0032**	0.0193	0.0046	0.0007	0.3750	0.5211	0.6446	0.7573	48.3233
epe_rp392	0.0025	0.0035	**0.0184**	**0.0036**	0.0006	0.3004	0.4663	0.5517	0.7541	47.0000
epe_p512	0.0045	0.0112	0.0856	0.0151	0.0007	0.2826	0.4519	0.5315	0.7298	46.1333
epe_s50t100	0.0027	0.0035	0.0186	0.0142	0.0002	0.3972	0.5366	0.6353	0.7693	46.6600
epe_s50t250	0.0048	0.0064	0.0462	0.0508	0.0027	0.4502	0.5858	0.6953	0.7886	47.2700
epe_s50t300	0.0028	0.0061	0.0320	0.0172	0.0009	0.4532	0.5823	0.6845	0.7996	49.0700
epe_s50t350	0.0050	0.0140	0.1232	0.0202	0.0020	0.3668	0.5056	0.6113	0.7423	47.3500
epe_s50t300_knn4	0.0024	0.0064	0.0555	0.0083	0.0007	0.3871	0.5224	0.6477	0.7744	**49.2000**
epe_s200t50	**0.0019**	0.0036	0.0208	0.0080	0.0004	0.4179	0.5527	0.6606	0.7660	48.9267
epe_s200t300	0.0041	0.0061	0.0407	0.0349	0.0016	0.4491	0.5782	0.6865	0.7910	47.1267
epe_s200t300_sem_t001	0.0031	0.0048	0.0324	0.0360	0.0029	0.4092	0.5461	0.6715	0.8061	47.6800
epe_s200t300_daw9	0.0041	0.0059	0.0355	0.0357	0.0009	0.4328	0.5683	0.6789	0.7696	48.4400

Table 5.8: Semantic and textural evaluation of all configurations. The configuration *epe_s50t300_knn4* achieves the best results. Combining larger sampling with semantic verification (*epe_s200t300_sem_t001*) does not beat the *epe* baseline. Domain adaptive weighting in combination with larger sampling (*epe_s200t300_daw9*) beats the *epe* baseline and more experiments have to be evaluated. *Six trainings were employed for the configuration *real_random* which sets the baseline for previous style-transfer GAN training on domain adaption with an average of 46.71 mAP and standard deviation of 0.90. **Three training were done for the configuration *epe* which sets the baseline for matching as proposed by Richter et al. [RAHK22] with an average of 48.05 mAP and a standard deviation of 0.43.

Chapter 6

Conclusion and Discussion

This thesis enhances the matching framework from Richter et al. [RAHK22] with sampling, matching and weighting methods to sample patches for style-transfer. Pixel accuracy[HTP+18] for semantic consistency and the sKVD [RAHK22] for perceptual alignment are used to evaluate style-transfer GANs with regards to the mAP. The best configuration *epe_s50t300_knn4* achieves 49.2 mAP in comparison to the previous baseline of 46.71 mAP for random sampling.

Experiments show that sampling and matching patches improves style-transfer and mAP. Matching both data domains with perceptual feature vectors extracted from a VGG16 [SZ15] network yield the best results for the downstream task as textures are style-transferred better. Aligning both domains based on high level semantic pseudo-labels improves the semantic consistency of style-transfer but does not improve object detection results since textures are aligned worse in comparison to the EPE[RAHK22] baseline. Filtering out VGG16 [SZ15] mismatches impacts style-transfer negatively regarding low perceptual semantics while it improves semantic consistency.

Sampling more patches on the target domain improves matching and enhances style-transfer of textures. Sampling more patches on the source domain improves matching, as more distinct matches are sampled for every iteration during training. However, merging both approaches caused the low-level perceptual alignment and mAP to decrease, which has to be analyzed further. Additionally, a good trade-off between compute time and performance has to be found, as the synthetic dataset with a size of approximately 6,500 images is smaller than synthetic datasets with 100,000 or more images. The largest configuration does take approximately 6 hours to generate 1,300,000 matches using FAISS[JDJ19] while smaller configurations can already improve style-transfer results. Shifting the dataset bias through domain adaptive sampling caused a degeneration of the GAN in experiments. Since GAN training is unstable and class distributions can be controlled

with weighting, this option was not further elaborated. Domain adaptive weighting improves the pixel accuracy of cars and generates minor light hallucinations which occur in the target dataset. The EPE[RAHK22] baseline thus was slightly beaten, though more experiments have to be performed for verification.

The patch size has major impact on GAN training as the GAN modifies more image content with larger patches. We employ 196 as the receptive field of the VGG16[SZ15] as standard patch size. The smaller patch size 128 causes the GAN to darken images which decreases semantic consistency, low-level perceptual alignment and thus mAP. In comparison to the patch size 196, the larger patch size 256 is slightly more semantically inconsistent, while low-level perceptual alignment and mAP improved. However, larger patch sizes than 256 hallucinate and become semantically inconsistent, which decreases mAP.

Future work can conduct more experiments with properties that improved mAP in comparison with the *epe* baseline [RAHK22], e.g. combinations of the configurations *epe_s50t300_knn4*, *epe_daw9* and *epe_rp256*. Results vary for different random seeds which makes evaluation more difficult because GAN training takes weeks. Thus, future work can evaluate more random seeds or more stable style-transfer models to get expressive results. CyCADA[HTP+18] for example employs additional semantic consistency constraints which could prevent configurations with semantic verification to degenerate. The sKVD[RAHK22] can be adapted to be evaluated on vehicles only, as mAP correlates with style-transfer of both whole images and downstream task relevant objects. As perceptual alignment of results in a higher mAP, better perceptual feature extractors can be employed. Therefore, recent advances for CNNs may improve perceptual description of the VGG16[SZ15] network [LMW+22]. Alternatively, fine-tuning is commonly done in image retrieval [CAS20] and perceptual metrics [ZIE+18] such that fine-tuning could also improve perceptual alignment.

Bibliography

[AGT+16] Relja Arandjelovic, Petr Gronat, Akihiko Torii, Tomas Pajdla, and Josef Sivic. Netvlad: Cnn architecture for weakly supervised place recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 5297–5307, 2016.

[BL15] Artem Babenko and Victor Lempitsky. Aggregating local deep features for image retrieval. In *Proceedings of the IEEE international conference on computer vision*, pages 1269–1277, 2015.

[BSAG18] Mikołaj Bińkowski, Danica J Sutherland, Michael Arbel, and Arthur Gretton. Demystifying mmd gans. In *International Conference on Learning Representations*, 2018.

[CAS20] Bingyi Cao, Andre Araujo, and Jack Sim. Unifying deep local and global features for image search. In *European Conference on Computer Vision*, pages 726–743. Springer, 2020.

[COR+16] Marius Cordts, Mohamed Omran, Sebastian Ramos, Timo Rehfeld, Markus Enzweiler, Rodrigo Benenson, Uwe Franke, Stefan Roth, and Bernt Schiele. The cityscapes dataset for semantic urban scene understanding. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3213–3223, 2016.

[DBK+21] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. In *International Conference on Learning Representations*, 2021.

[DDS+09] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image

database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009.

[EVGW+10] Mark Everingham, Luc Van Gool, Christopher KI Williams, John Winn, and Andrew Zisserman. The pascal visual object classes (voc) challenge. *International journal of computer vision*, 88(2):303–338, 2010.

[GBR+12] Arthur Gretton, Karsten M Borgwardt, Malte J Rasch, Bernhard Schölkopf, and Alexander Smola. A kernel two-sample test. *The Journal of Machine Learning Research*, 13(1):723–773, 2012.

[GPAM+14] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. *Advances in neural information processing systems*, 27, 2014.

[HDVG21] Lukas Hoyer, Dengxin Dai, and Luc Van Gool. Daformer: Improving network architectures and training strategies for domain-adaptive semantic segmentation. *arXiv preprint arXiv:2111.14887*, 2021.

[HRU+17] Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. Gans trained by a two time-scale update rule converge to a local nash equilibrium. *Advances in neural information processing systems*, 30, 2017.

[HTP+18] Judy Hoffman, Eric Tzeng, Taesung Park, Jun-Yan Zhu, Phillip Isola, Kate Saenko, Alexei Efros, and Trevor Darrell. Cycada: Cycle-consistent adversarial domain adaptation. In *International conference on machine learning*, pages 1989–1998. PMLR, 2018.

[HZRS16] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[JC12] Hervé Jégou and Ondřej Chum. Negative evidences and cooccurences in image retrieval: The benefit of pca and whitening. In *European conference on computer vision*, pages 774–787. Springer, 2012.

[JDJ19] Jeff Johnson, Matthijs Douze, and Hervé Jégou. Billion-scale similarity search with gpus. *IEEE Transactions on Big Data*, 2019.

[JDS10] Hervé Jégou, Matthijs Douze, and Cordelia Schmid. Product quantization for nearest neighbor search. *IEEE transactions on pattern analysis and machine intelligence*, 33(1):117–128, 2010.

[JRBM+17] Matthew Johnson-Roberson, Charles Barto, Rounak Mehta, Sharath Nittur Sridhar, Karl Rosaen, and Ram Vasudevan. Driving in the matrix: Can virtual worlds replace human-generated annotations for real world tasks? In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pages 746–753. IEEE, 2017.

[KSH12] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, 25:1097–1105, 2012.

[LKL+20] Guangrui Li, Guoliang Kang, Wu Liu, Yunchao Wei, and Yi Yang. Content-consistent matching for domain adaptive semantic segmentation. In *European Conference on Computer Vision*, pages 440–456. Springer, 2020.

[LLC+21] Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. Swin transformer: Hierarchical vision transformer using shifted windows. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 10012–10022, 2021.

[LLS+20] John Lambert, Zhuang Liu, Ozan Sener, James Hays, and Vladlen Koltun. Mseg: A composite dataset for multi-domain semantic segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2879–2888, 2020.

[LMB+14] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *European conference on computer vision*, pages 740–755. Springer, 2014.

[LMW+22] Zhuang Liu, Hanzi Mao, Chao-Yuan Wu, Christoph Feichtenhofer, Trevor Darrell, and Saining Xie. A convnet for the 2020s. *arXiv preprint arXiv:2201.03545*, 2022.

[NAS+17] Hyeonwoo Noh, Andre Araujo, Jack Sim, Tobias Weyand, and Bohyung Han. Large-scale image retrieval with attentive deep

local features. In *Proceedings of the IEEE international conference on computer vision*, pages 3456–3465, 2017.

[NBTM20] Tony Ng, Vassileios Balntas, Yurun Tian, and Krystian Mikolajczyk. Solar: second-order loss and attention for image retrieval. In *European Conference on Computer Vision*, pages 253–270. Springer, 2020.

[RAHK22] Stephan R Richter, Hassan Abu Al Haija, and Vladlen Koltun. Enhancing photorealism enhancement. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2022.

[RHGS15] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. *Advances in neural information processing systems*, 28, 2015.

[RTC18] Filip Radenović, Giorgos Tolias, and Ondřej Chum. Fine-tuning cnn image retrieval with no human annotation. *IEEE transactions on pattern analysis and machine intelligence*, 41(7):1655–1668, 2018.

[RVRK16] Stephan R Richter, Vibhav Vineet, Stefan Roth, and Vladlen Koltun. Playing for data: Ground truth from computer games. In *European conference on computer vision*, pages 102–118. Springer, 2016.

[SSSG17] Chen Sun, Abhinav Shrivastava, Saurabh Singh, and Abhinav Gupta. Revisiting unreasonable effectiveness of data in deep learning era. In *Proceedings of the IEEE international conference on computer vision*, pages 843–852, 2017.

[SVI+16] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2818–2826, 2016.

[SZ15] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. In *International Conference on Learning Representations*, 2015.

[TJC20] Giorgos Tolias, Tomas Jenicek, and Ondřej Chum. Learning and aggregating deep local descriptors for instance-level recognition. In *European Conference on Computer Vision*, pages 460–477. Springer, 2020.

[TSJ16] Giorgos Tolias, Ronan Sicre, and Hervé Jégou. Particular object retrieval with integral max-pooling of cnn activations. In *ICLR 2016-International Conference on Learning Representations*, pages 1–12, 2016.

[WEG87] Svante Wold, Kim Esbensen, and Paul Geladi. Principal component analysis. *Chemometrics and intelligent laboratory systems*, 2(1-3):37–52, 1987.

[WLZ+18] Ting-Chun Wang, Ming-Yu Liu, Jun-Yan Zhu, Andrew Tao, Jan Kautz, and Bryan Catanzaro. High-resolution image synthesis and semantic manipulation with conditional gans. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 8798–8807, 2018.

[YUC+19] Jaejun Yoo, Youngjung Uh, Sanghyuk Chun, Byeongkyu Kang, and Jung-Woo Ha. Photorealistic style transfer via wavelet transforms. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 9036–9045, 2019.

[ZF14] Matthew D Zeiler and Rob Fergus. Visualizing and understanding convolutional networks. In *European conference on computer vision*, pages 818–833. Springer, 2014.

[ZIE+18] Richard Zhang, Phillip Isola, Alexei A Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 586–595, 2018.

[ZPIE17] Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. In *Proceedings of the IEEE international conference on computer vision*, pages 2223–2232, 2017.

[ZZZ+21] Pan Zhang, Bo Zhang, Ting Zhang, Dong Chen, Yong Wang, and Fang Wen. Prototypical pseudo label denoising and target structure learning for domain adaptive semantic segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12414–12424, 2021.

Chapter 7

Appendix

Figure 7.1: Matching patches of the configuration *epe_s50t350* from the GTA dataset *Driving in the Matrix* (DiM) [JRBM+17] with *Cityscapes*(CS) [COR+16] patches based on VGG16 [SZ15] features. Matches lie in a ϕ_{VGG} cosine similarity range $[0.95; 1.0]$ and in a ϕ_{SEM} cosine similarity range $[0.9; 1.0]$.

Figure 7.2: Matching patches of the configuration *epe_s50t350* from the GTA dataset *Driving in the Matrix* (DiM) [JRBM+17] with *Cityscapes* (CS) [COR+16] patches based on VGG16 [SZ15] features. Matches lie in a ϕ_{VGG} cosine similarity range $[0.5; 0.6]$ and in a ϕ_{SEM} cosine similarity range $[0.0; 1.0]$.

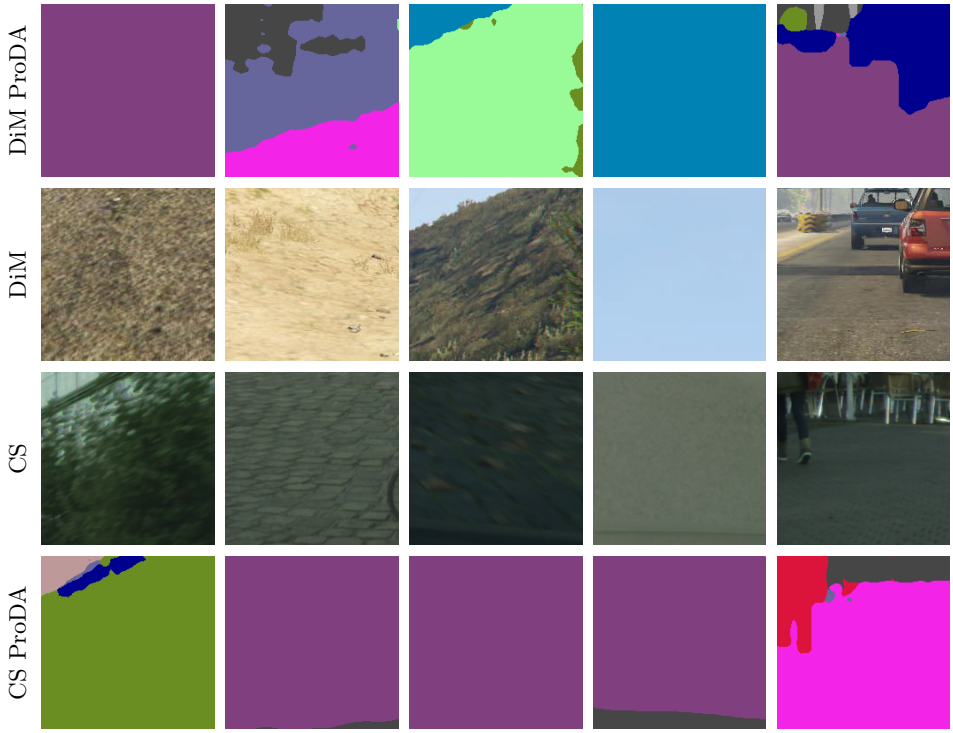


Figure 7.3: Matching patches of the configuration *epe_s50t350* from the GTA dataset *Driving in the Matrix* (DiM) [JRBM+17] with *Cityscapes* (CS) [COR+16] patches based on VGG16 [SZ15] features. Semantic segmentation pseudo-labels are extracted with ProDA [ZZZ+21]. Matches lie in a ϕ_{VGG} cosine similarity range $[0.9; 1.0]$ and in a ϕ_{SEM} cosine similarity range $[0.0; 0.10]$.

Figure 7.4: Matching patches of the configuration *epe_p256* (patch size 256 with GeM[RTC18] pooling)from the GTA dataset *Driving in the Matrix* (DiM) [JRBM+17] with *Cityscapes* (CS) [COR+16] patches based on VGG16 [SZ15] features. Matches lie in a ϕ_{VGG} cosine similarity range $[0.85; 0.95]$ and in a ϕ_{SEM} cosine similarity range $[0.8; 1.0]$.

Figure 7.5: Matching patches of the configuration *epe_p256* (patch size 256 with GeM[RTC18] pooling)from the GTA dataset *Driving in the Matrix* (DiM) [JRBM+17] with *Cityscapes* (CS) [COR+16] patches based on VGG16 [SZ15] features. Matches lie in a ϕ_{VGG} cosine similarity range $[0.5; 0.6]$ and in a ϕ_{SEM} cosine similarity range $[0.0; 1.0]$.

Figure 7.6: Matching patches of the configuration *epe_rp392* (patch size 392 with resizing) from the GTA dataset *Driving in the Matrix* (DiM) [JRBM+17] with *Cityscapes* (CS) [COR+16] patches based on VGG16 [SZ15] features. Matches lie in a ϕ_{VGG} cosine similarity range $[0.95; 1.0]$ and in a ϕ_{SEM} cosine similarity range $[0.8; 1.0]$.

Figure 7.7: Matching patches of the configuration *epe_rp392* (patch size 392 with resizing) from the GTA dataset *Driving in the Matrix* (DiM) [JRBM+17] with *Cityscapes* (CS) [COR+16] patches based on VGG16 [SZ15] features. Matches lie in a ϕ_{VGG} cosine similarity range $[0.5; 0.6]$ and in a ϕ_{SEM} cosine similarity range $[0.0; 1.0]$.