ACE: Adapting to Changing Environments for Semantic Segmentation

Zuxuan Wu[1], Xin Wang[2], Joseph E. Gonzalez[2], Tom Goldstein[1], Larry S. Davis[1]

[1]University of Maryland, [2]UC Berkeley

Abstract

Deep neural networks exhibit exceptional accuracy when they are trained and tested on the same data distributions. However, neural classifiers are often extremely brittle when confronted with domain shift—changes in the input distribution that occur over time. We present ACE, a framework for semantic segmentation that dynamically adapts to changing environments over time. By aligning the distribution of labeled training data from the original source domain with the distribution of incoming data in a shifted domain, ACE synthesizes labeled training data for environments as it sees them. This stylized data is then used to update a segmentation model so that it performs well in new environments. To avoid forgetting knowledge from past environments, we introduce a memory that stores feature statistics from previously seen domains. These statistics can be used to replay images in any of the previously observed domains, thus preventing catastrophic forgetting. In addition to standard batch training using stochastic gradient decent (SGD), we also experiment with fast adaptation methods based on adaptive meta-learning. Extensive experiments are conducted on two datasets from SYNTHIA, the results demonstrate the effectiveness of the proposed approach when adapting to a number of tasks.

1. Introduction

When computer vision systems are deployed in the real world, they are exposed to changing environments and non-stationary input distributions that pose major challenges. For example, a deep network optimized using images collected on sunny days with clear skies may fail drastically at night under different lighting conditions. In fact, it has been recently observed that deep networks demonstrate severe instability even under small changes to the input distribution [12], let alone when confronted with dynamically changing streams of information.

The problem of domain shift can be avoided by collecting sufficient training data to cover all possible input distributions that occur at test time. However, the expense of collecting and manually annotating data makes this infeasible in many applications. This is particularly true for detailed visual understanding tasks like object detection and semantic segmentation where image annotation is labor-intensive. It is worth noting that humans are capable of "lifelong learning," in which new tasks and environments are analyzed using accumulated knowledge from the past. However, achieving the same goal in deep neural networks is non-trivial as (i) new data domains come in at real time without labels, and (ii) deep networks suffer from *catastrophic forgetting* [33], in which performance drops on previously learned tasks when optimizing for new tasks.

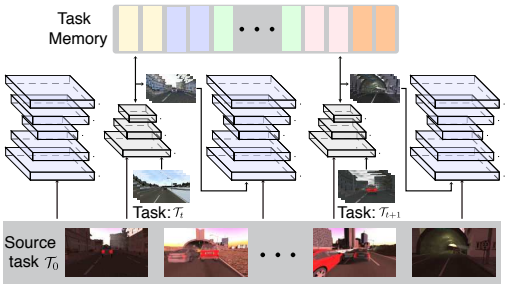

Figure 1: **A conceptual overview of the framework**. ACE adapts a model trained on a source task to a sequence of target tasks. This is done by aligning the feature statistics of labeled images from a source task with incoming images of the target task. This alignment produces labeled images in the target domain that can be used to update the segmentation model. A memory unit also facilitates replay of images from domains seen in the past to prevent forgetting.

We consider the lifelong learning problem of adapting a pre-trained model to dynamically changing environments, whose distributions reflect disparate lighting and weather conditions. In particular, we assume access to image-label pairs from an original *source* environment, and only unlabeled images from new *target* environments that are not observed in the training data. Furthermore, we consider the difficulties posed by learning over time, in which target environments appear sequentially.

We focus on the specific task of semantic segmenta-

tion due to its practical applications in autonomous driving, where a visual recognition system is expected to deal with changing weather and illumination conditions. This application enables us to leverage the convenience of collecting data from different distributions using graphic rendering tools [43, 42].

To this end, we introduce ACE, a framework which adapts a pre-trained segmentation model to a stream of new tasks that arrive in a sequential manner, while storing historical style information in a compact memory to avoid forgetting (see Figure 2 for an overview). In particular, given a new task, we use an image generator to align the distribution of (labeled) source data with the distribution of (unlabeled) incoming target data at the pixel-level. This produces labeled images with color and texture properties that closely reflect the target domain, which are then directly used for training the segmentation network on the new target domain. Style transfer is achieved by renormalizing feature maps of source images so they have first- and second-order feature statistics that match target images [19, 60]. These renormalized feature maps are then fed into a generator network that produces stylized images.

What makes ACE unique is its ability to learn over a lifetime. To prevent forgetting, ACE contains a compact and light-weight memory that stores the feature statistics of different styles. These statistics are sufficient to regenerate images in any of the historical styles without the burden of storing a library of historical images. Using the memory, historical images can be re-generated and used for training throughout time, thus stopping the deleterious effects of catastrophic forgetting. The entire generation and segmentation framework can be trained in a joint end-to-end manner with SGD. Finally, we consider the topic of using adaptive meta-learning to facilitate faster adaptation to new environments when they are encountered.

Our main contributions are summarized below: (1) we present a lightweight framework for semantic segmentation, which is able to adapt to a stream of incoming distributions using simple and fast optimization; (2) we introduce a memory that stores feature statistics for efficient style replay, which facilitates generalization on new tasks without forgetting learned knowledge; (3) we consider meta-learning strategies to speed up the rate of adaptation to new problem domains; (4) we conduct extensive experiments on two subsets of SYNTHIA [44] and the experiments demonstrate the effectiveness of our method when adapting to a sequence of tasks with different weather and lighting conditions.

2. Related Work

Unsupervised Domain Adaptation. Our work relates to unsupervised domain adaptation, which aims to improve the generalization of a pre-trained model when testing on novel distributions without accessing labels. Existing methods along this line of research aim to reduce domain differences at either the feature or pixel level. In particular, feature-level adaptation focuses on aligning features used for the target task (*e.g.*, classification or segmentation) by minimizing a notion of distance between source and target domains. Such notion of distance can be explicit metrics in the forms of Maximum Mean Discrepancies (MMD) [31, 4], covariances [54], *etc.*; or implicitly estimated to make features domain-invariant using adversarial loss functions such as reversed gradient [8, 9], domain confusion [57], or Generative Adversarial Network [58, 16, 17, 45, 18], *etc*.

On the other hand, pixel-level adaptation transforms images from different domains to look as if they were drawn from the same distribution by using a mapping that reduces inconsistencies in texture and lighting [3, 52, 55, 29]. There are also recent methods seeking to align both pixel-level and feature-level representations simultaneously [15, 62, 69]. In addition, Zhang *et al*. present a curriculum strategy that uses global label distributions and local super-pixel distributions for adaptation [68]. Saleh *et al*. handle foreground classes using detection methods when addressing domain shift [46]. Our framework differs from previous work as we are adapting to a stream of testing domains that arrive sequentially rather than a single fixed one, which is challenging as it requires the network to perform well on both current and all previous domains. Note that although we mainly focus on pixel-level alignment, our method can further benefit from feature-level alignment in the segmentation network, but at the cost of saving raw images as opposed to only feature statistics. Further, our approach is also related to [63, 2, 14] that perform sequential adaptation for classification tasks by aligning at feature-level, while ours focuses on semantic segmentation with alignment at pixel-level.

Image Synthesis and Stylization. There is a growing interest in synthesizing images with Generative Adversarial Networks (GANs) [65, 38, 29], which is formulated as a minimax game between a generator and a discriminator [11]. To control the generation process, a multitude of additional information has been incorporated including labels [36], text [41], attributes [49], and images [21, 25]. GANs have also been used in the context of image-to-image translation, which transfers the style of an image to that of a reference image using either cycle-consistency [71] or mapping into a shared feature space [28, 20]. Without knowing joint distributions of domains, these approaches attempt to learn conditional distributions with marginal distributions from each domain. However, generating high resolution images with GANs still remains a difficult problem and is computationally intensive [23]. In contrast, methods for neural style transfer [10, 19, 59, 37, 22] usually avoid the difficulties of generative modeling, and simply match the feature statistics of Gram matrices [10, 22] or perform channel-wise alignment of mean and variance [27, 19]. In our work, we build

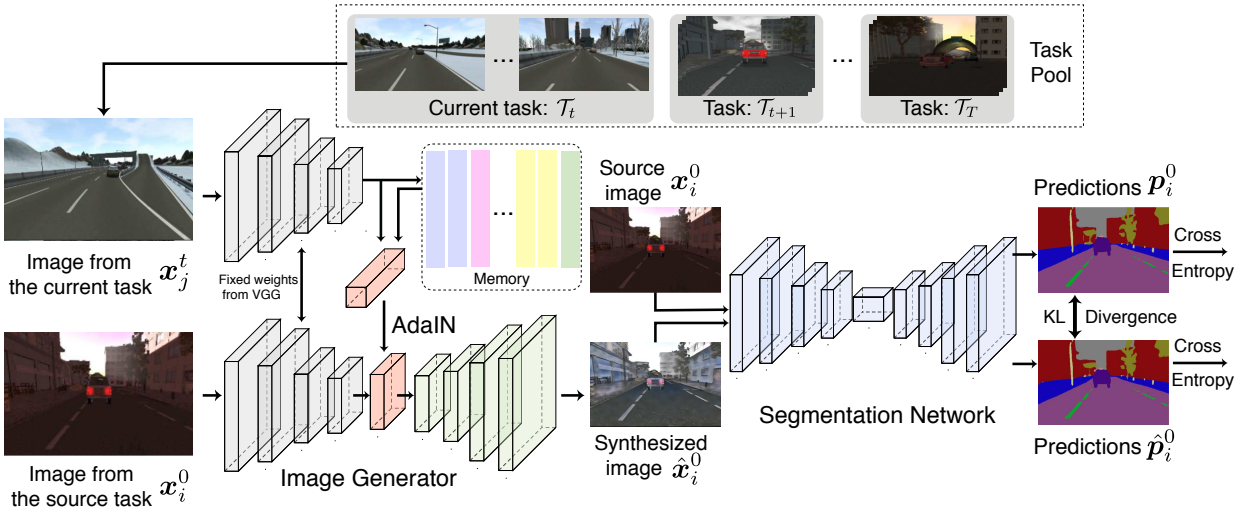

Figure 2: **An overview of the proposed framework**. Given an incoming task, ACE synthesizes new images that preserve the contents of images from the source task but in the style of a target task. This is done either by transferring style information from incoming images onto the source images, or by sampling style information from the memory unit. With these synthesized images in different styles, the segmentation network is trained to generalize to new tasks without forgetting knowledge learned in the past.

upon style transfer to synthesize new images in the style of images from the current task while preserving the contents of the source image.

Lifelong Learning. Our work is also related to lifelong learning, or continual learning, which learns progressively and adapts to new tasks using knowledge accumulated throughout the past. Most existing work focuses on mitigating catastrophic forgetting when learning new tasks [24, 67, 40, 50, 51, 32, 5]. Several recent approaches propose to dynamically increase model capacities when new tasks arrive [66, 64]. Our work focuses on how to adapt a learned segmentation model in an unsupervised manner to a stream of new tasks, each with image distributions different from those originally used for training. In addition, to avoid forgetting knowledge learned in the past, styles are represented and cataloged using their feature statistics. Because this representation is much smaller than raw images, the framework is scalable.

Meta-Learning. Meta-learning [48, 56], also known as learning to learn, is a setting where an agent ingests a set of tasks, each a learning problem on its own, and then establishes a model that can be quickly adapted to unseen tasks from the same distribution. There are three categories of meta-learners: (i) model-based with an external memory [47, 34]; (ii) metric-based [61]; and (iii) optimization-based [7, 35]. Existing approaches mainly focus on few shot classification, regression, and reinforcement learning problems, while our approach focuses on how to adapt segmentation models efficiently.

3. Approach

The goal of ACE is to adapt a segmentation model from a source task to a number of sequentially presented target tasks with disparate image distributions. The method transfers labeled source images into target domains to create synthetic training data for the segmentation model, while memorizing style information to be used for style replay to prevent forgetting.

More formally, let $\mathcal{T}_0$ denote the source task and $\{\mathcal{T}_i\}_{i=0}^{T}$ represent T target tasks that arrive sequentially. We further use $\boldsymbol{X}^0 = \{(\boldsymbol{x}_1^0, \boldsymbol{y}_1^0), \cdots, (\boldsymbol{x}_N^0, \boldsymbol{y}_N^0)\}$ [1] to represent the N images and their corresponding labels used for the source task. The label $\boldsymbol{y}_i^0$ contains a one-hot label vector for each pixel in the image $\boldsymbol{x}_i^0$; we denote the i-th image sample as $\boldsymbol{x}_i^0 \in \mathbb{R}^{3\times H\times W}$, and $\boldsymbol{y}_i^0 \in \{0,1\}^{C\times H\times W}$ represents the corresponding label maps, with H and W being the height and width respectively and C denoting the number of classes.

For each subsequent target task, we assume access to only images rather than image-label pairs as in the source task. We further denote the number of target tasks as T and use $\boldsymbol{X}^t = \{\boldsymbol{x}_1^t, \cdots, \boldsymbol{x}_{N^t}^t\}$ for $t \in [1 \cdots T]$ to represent the image set for the t-th incoming task, which has N^t images of the same resolution as the source data.

ACE contains four key components: an encoder, a generator, a memory, and a segmentation network. The encoder network converts a source image $\boldsymbol{x}_i^0$ into a feature representation $\boldsymbol{z}_i^0$ (in our case, a stack of 512 output feature maps).

[1] We omit $\mathcal{T}$ here for the ease of notation.

The generator network converts feature representations $\boldsymbol{z}$ into images. The style of the resulting image can be controlled/manipulated by modifying the statistics (mean and standard deviation of each feature map) of $\boldsymbol{z}$ before it is handed to the generator. The memory unit remembers the feature statistics (1024 scalar values per style, corresponding to the mean and standard deviation of each of the 512 feature maps) for each image style/domain. A source image can be stylized into any previously seen domain by retrieving the relevant style statistics from the memory unit, renormalizing the feature maps of the source image to have the corresponding statistics, and then handing the renormalized features to the generator to create an image.

Stylization via the encoder and generator. When a new task is presented, labeled images are created in the new task domain by transferring source images (and their accompanying labels) to the target domain. To do this, we jointly train a generator network for producing target-stylized images, and a segmentation network for processing images in the target domain.

The image generation pipeline begins with an encoder that extracts feature maps from images. We use a pre-trained VGG19 network [53] as the encoder, taking the output from `relu4` to define f_{enc}. Following [26, 19], the weights of the encoder are frozen during training to extract fixed representations $f_{enc}(\boldsymbol{x}_i^0) = \boldsymbol{z}_i^0$ and $f_{enc}(\boldsymbol{x}_j^t) = \boldsymbol{z}_j^t$ from images $\boldsymbol{x}_i^0$ and $\boldsymbol{x}_j^t$, respectively.

The image generator f_{gen}, parameterized by weights $\boldsymbol{\psi}_{gen}$, de-convolves feature maps into images. The style of the output image can be borrowed from a target image with AdaIN [19], which re-normalizes the feature maps (*i.e.*, channels) $\boldsymbol{z}_i^0$ of source images to have the same mean and standard deviation as the maps of a selected target image $\boldsymbol{z}_i^t$:

$$\hat{\boldsymbol{z}}_i^0 = \text{AdaIN}(\boldsymbol{z}_i^0, \boldsymbol{z}_j^t) = \sigma(\boldsymbol{z}_j^t)\frac{\boldsymbol{z}_i^0 - \mu(\boldsymbol{z}_i^0)}{\sigma(\boldsymbol{z}_i^0)} + \mu(\boldsymbol{z}_j^t). \quad (1)$$

Here, $\sigma(\boldsymbol{z})$ and $\mu(\boldsymbol{z})$ compute the mean and variance of each channel of $\boldsymbol{z}$, respectively. The normalized feature maps $\hat{\boldsymbol{z}}_i^0$ can be stuffed into the generator to synthesize a new image $\hat{\boldsymbol{x}}_i^0 = f_{gen}(\hat{\boldsymbol{z}}_i^0; \boldsymbol{\psi}_{gen})$. If the parameters $\boldsymbol{\psi}_{gen}$ are properly tuned, the resulting image will have the contents of $\boldsymbol{x}_i^0$ but in the style of $\boldsymbol{x}_j^t$.

We train the generator so that it acts as an inverse for the encoder; the encoder should map the decoded image (approximately) onto the features that produced it. We enforce this by minimize the following loss function:

$$\begin{aligned}\ell_{gen}(\boldsymbol{\psi}_{gen}) =& \|\tilde{\boldsymbol{z}} - \hat{\boldsymbol{z}}_i^0\|_2 + \|\mu(\tilde{\boldsymbol{z}}) - \mu(\boldsymbol{z}_j^t)\|_2 \\ &+ \|\sigma(\tilde{\boldsymbol{z}}) - \sigma(\boldsymbol{z}_j^t)\|_2, \\ \text{where } \tilde{\boldsymbol{z}} &= f_{enc}(f_{dec}(\hat{\boldsymbol{z}}_i^0; \boldsymbol{\psi}_{gen})).\end{aligned} \quad (2)$$

Here, the first term (the content loss) measures the differences between features of the generated image and the aligned features of the source image with an aim to preserve the contents the source images. The remaining two terms force the generated image into the style of $\boldsymbol{x}_j^t$ by matching the mean and variance of feature maps per-channel. Note that some authors match Gram matrices [10, 62] to make styles consistent. We match mean and variances of feature maps as in [27, 59] since these statistics are simple to optimize and contain enough information to get a good stylization. In contrast to using several layers for alignment [27, 19], we simply match one layer of feature maps from the VGG encoder, which is faster yet sufficient. More importantly, this facilitates lightweight style replay as will be described below.

The segmentation network. The newly synthesized image $\hat{\boldsymbol{x}}_i^0 = f_{dec}(\tilde{\boldsymbol{z}}; \boldsymbol{\psi}_{gen})$ is handed to the segmentation network f_{seg}, parameterized by weights $\boldsymbol{\psi}_{seg}$. This network produces a map of label vectors $\hat{\boldsymbol{p}}_i^0 = f_{seg}(\hat{\boldsymbol{x}}_i^0; \boldsymbol{\psi}_{seg})$, and is trained by minimizing a multi-class cross-entropy loss summed over pixels. In addition, since the synthesized image might lose certain details of the original images that could degrade the performance of the segmentation network, we further constrain outputs of the synthetic image $\hat{\boldsymbol{x}}_i^0$ from the segmentation network $\hat{\boldsymbol{p}}_i^0$ to be close to the predictions $\boldsymbol{p}_i^0$ of the original image $\boldsymbol{x}_i^0$ before stylization. This is achieved by measuring the `KL`-divergence between these two outputs, which is similar in spirit to knowledge distillation [13] with the outputs from the original image serving as the teacher. The segmentation loss takes the following form:

$$\begin{aligned}\ell_{seg}(\boldsymbol{\psi}_{seg}) = &- \sum_{m=1}^{H\times W} \texttt{KL}\left(\hat{\boldsymbol{p}}_{i,m}^0 \,\middle\|\, \boldsymbol{p}_{i,m}^0\right) \\ &+ \sum_{c=1}^{C} \boldsymbol{y}_{i,mc}^0 \texttt{log}(\hat{\boldsymbol{p}}_{i,mc}^0).\end{aligned} \quad (3)$$

Finally, combining Eqn. 2 and Eqn. 3, we have the following objective function:

$$\mathcal{L}(\boldsymbol{\psi}) = \mathop{\mathbb{E}}_{\substack{(\boldsymbol{x}^0, \boldsymbol{y}^0)\sim \boldsymbol{X}^0 \\ \boldsymbol{x}^t \sim \boldsymbol{X}^t}} \ell_{gen}(\boldsymbol{x}^0, \boldsymbol{x}^t) + \ell_{seg}(\boldsymbol{x}^0, \boldsymbol{y}^0, \boldsymbol{x}^t), \quad (4)$$

where $\boldsymbol{\psi} = \{\boldsymbol{\psi}_{seg}, \boldsymbol{\psi}_{gen}\}$ denotes the parameters of the network. Note that the segmentation loss implicitly depends on the generator parameters because segmentation is performed on the output of the generator.

Memory unit and style replay. Optimizing Eqn 4 reduces the discrepancies between the source task and the target task, yet it is unclear how to continually adapt the model to a sequence of incoming tasks containing potentially different image distributions without forgetting knowledge learned in

the past. A simple way is to store a library of historical images from previous tasks, and then randomly sample images from the library for replay when learning new tasks. However, this requires large working memory which might not be viable, particularly for segmentation tasks, where images are usually of high resolutions (*e.g.*, 1024×2048 for images in Cityscapes [42]).

Fortunately, the proposed alignment process in Eqn. 1 synthesizes images from the target distribution using only a source image, and the mean and variance of each channel in the feature maps $\boldsymbol{z}_j^t$ from a target image. Therefore, we only need to save the feature statistics (512-D for both mean and variance) in the memory $\mathcal{M}$ for efficient replay. When learning the t-th task $\mathcal{T}_t$, we select a sample of test images and store their 1024-D feature statistics in the memory. When adapting to the next task $\mathcal{T}_{t+1}$, in addition to sampling from $\boldsymbol{X}^{t+1}$, we also randomly access the memory $\mathcal{M}$, which contains style information from previous tasks, to synthesize images that resemble seen tasks on-the-fly for replay.

Faster adaptation via adaptive meta-learning. Recent methods in meta-learning [7, 35, 39] produce flexible models having meta-parameters with the property that they can be quickly adapted to a new task using just a few SGD updates. While standard SGD offers good performance when optimizing Eqn. 4 for a sufficient number of epochs, we now explore whether adaptive meta-learning can produce models that speed up adaptation.

For this purpose, we use Reptile [35], which is an inexpensive approximation of the MAML [7] method. Reptile updates parameters of a meta-model by first selecting a task at random, and performing multiple steps of SGD to fine-tune the model for that task. Then a "meta-gradient" step is taken in the direction of the fine-tuned parameters. The next iteration proceeds with a different task, and so on, to generate a meta-model with parameters that are only a small perturbation away from the optimal parameters for a wide range of tasks.

To be precise, the Reptile meta gradient $\mathbf{g}_t(\boldsymbol{\psi})$ is defined as:

$$\mathbf{g}_t(\boldsymbol{\psi}) = \boldsymbol{\psi}_t - \widetilde{\boldsymbol{\psi}}, \text{ where } \widetilde{\boldsymbol{\psi}} = U^k(\boldsymbol{\psi}_t). \tag{5}$$

Here $U^k(\boldsymbol{\psi}_t)$ denotes k steps of standard SGD for a randomly selected task. To achieve fast adaptation, we sample from the current task as well as the memory to perform meta-updates using meta-gradients from the whole history of tasks. The meta-gradients are then fine-tuned on the current task to evaluate performance. The algorithm is summarized in Alg. 1.

Algorithm 1 Fast Adaptation with Adaptive Meta-Learning

Input: $\boldsymbol{X}^0 = \{(\boldsymbol{x}_1^0, \boldsymbol{y}_1^0), \cdots, (\boldsymbol{x}_N^0, \boldsymbol{y}_N^0)\}$
A pre-trained segmentation model, whose parameters are $\boldsymbol{\psi}$
The memory is initialized as empty $\mathcal{M} \leftarrow [\,]$
for $t = 1, \ldots T$ **do**
  initialize $\mathcal{D}_t = \emptyset$
  while $|\mathcal{D}_t| < N^t$ **do**
    for `iterations` $= 1, \ldots, I$ **do**
      Append batch of n image samples $\boldsymbol{x}^t$ to $\mathcal{D}_t$
      Sample batches of $(\boldsymbol{x}^s, \boldsymbol{y}^s)$ from the source task
      Sample batches of $\boldsymbol{x}^t$ from the t-th task
      Compute $\boldsymbol{z}^t$ with the VGG encoder using sampled $\boldsymbol{x}^t$
      Sample seen tasks $\boldsymbol{z}^l (l < t)$ from the memory $\mathcal{M}$
      $\boldsymbol{\psi}_t \leftarrow \boldsymbol{\psi}_t - \eta\, \mathbf{g}_t(\boldsymbol{\psi})$, $\mathbf{g}_t(\boldsymbol{\psi})$ are computed with Eqn. 5
      Update $\mathcal{M}$ by storing some randomly selected $\boldsymbol{z}^t$
    end for
    $\widetilde{\boldsymbol{\psi}}_t \leftarrow \boldsymbol{\psi}_t - \alpha \nabla \mathcal{L}(\boldsymbol{\psi}_t; \mathcal{D}_t)$ ▷ For testing the t-th task
  end while
  $\boldsymbol{\psi}_{t+1} \leftarrow \boldsymbol{\psi}_t$
end for

4. Experiments

In this section, we first introduce the experimental setup and then we report results of ACE and provide discussions.

4.1. Experimental Setup

Datasets and evaluation metrics. Since our approach is designed to process different input distributions sharing the same label space for segmentation tasks, we use data with various weather and lighting conditions from SYNTHIA [44], a large-scale synthetic dataset generated with rendering engines for semantic segmentation of urban scenes. We use SYNTHIA-SEQS, a subset of SYNTHIA showing the viewpoint of a virtual car captured across different seasons. This dataset can be broken down into various weather and illumination conditions including "summer", "winter", "rain", "winter-night", *etc*. (See Table 1). We consider two places from SYNTHIA-SEQS for evaluation, HIGHWAY and NYC-LIKE CITY, which contain 9 and 10 video sequences with different lighting conditions, respectively. We treat each sequence as a task, with around $1,000$ images on average, and each task is further split evenly into a training set and a validation set.

We first train a segmentation model using labeled images in the "dawn" scenario, and then adapt the learned model to the remaining tasks in each of the sequences in an unsupervised setting [2]. During the adaptation process, following [68, 16], we only access labeled images from the first task (*i.e.*, "dawn"), and unlabeled images from the current task. To evaluate the performance of the segmentation model, we report mean intersection-over-union (mIoU) on the validation set of each task as well as the mean mIoU across all tasks.

Network architectures. We use a pretrained VGG19 network as the encoder, and the architecture of the decoder is detailed in the supplemental material. We evaluate the performance of our framework with three different

[2] We use "dawn" as the source domain since it comes first alphabetically in all domains.

Method	Arch.	HIGHWAY											NYC-LIKE CITY											
		Dawn	Fall	Fog	Night	Spring	Summer	Sunset	Winter	WinterNight	mean mIOU	gain	Dawn	Fall	Fog	Night	RainNight	SoftRain	Spring	Summer	Sunset	Winter	mean mIOU	gain
Source	A	65.4	61.4	62.3	59.4	62.3	62.1	64.9	50.0	54.5	60.2	-	46.4	42.0	41.0	37.9	30.2	32.0	42.5	40.9	41.0	38.6	39.3	-
ACE	A	67.8	65.0	65.4	62.8	65.4	64.8	66.8	55.5	58.5	63.6	3.4	53.9	50.7	52.3	50.2	40.4	42.4	50.6	51.5	52.1	44.5	48.9	9.6
Source	B	68.9	53.4	50.5	39.2	59.2	59.3	62.5	39.5	32.6	51.7	-	57.7	24.0	25.9	20.8	13.9	15.1	39.1	34.5	36.2	21.6	28.9	-
ACE	B	69.6	65.3	66.2	63.9	66.5	66.7	69.2	53.7	59.0	64.5	12.8	55.8	51.6	51.7	49.8	43.5	48.6	52.7	51.1	52.8	46.0	50.4	21.5
Source	C	68.3	66.1	66.0	58.2	66.4	65.8	68.3	53.4	53.2	62.8	-	57.3	50.6	51.4	47.2	36.4	39.0	53.2	52.2	53.1	43.6	48.4	-
ACE	C	70.7	69.5	69.8	67.9	69.1	68.5	70.9	59.4	63.7	67.7	4.93	58.5	56.1	55.9	54.2	42.6	46.1	55.6	56.4	56.6	50.8	53.3	4.9

Table 1: **Results of different backbone networks used for adapting to changing environments.** Here, "Source" denotes directly applying the segmentation model to target tasks without adaptation. A, B, C represent FCN-8s-ResNet101, DeepLab V3+ [6], and ResNet50-PSPNet [70], respectively.

segmentation architectures, FCN-8s-ResNet101, DeepLab V3+ [6], and ResNet50-PSPNet [70], which have demonstrated great success on standard benchmarks. FCN-8s-ResNet101 is an extension of FCN-8s-VGG network [30] that uses ResNet101 with dilations as the backbone, rather than VGG19. ResNet50-PSPNet contains a pyramid pooling module to derive representations at different levels that encompass sufficient context information [70]. DeepLab V3+ [6] introduces a decoder to refine the segmentation results along object boundaries.

Implementation details. We use PyTorch for implementation and use SGD as the optimizer with a weight decay of 5×10^{-5} and a momentum of 0.99. We set the learning rate to 10^{-3} and optimize for 10000 iterations using standard SGD for training both source and target tasks. For fast adaptation with meta-gradients, we perform 50 steps of meta updates. We sample three source images in a mini-batch for training, and for each of these images from the source task we randomly sample two reference images, one from the current target task and one from the memory, as style references for generating new images. For style replay, the memory caches 100 feature vectors per task representing style information from 100 target images.

4.2. Results and Discussion

Effectiveness of adapting to new tasks. Table 1 presents the results of ACE and comparisons with source only methods, which directly apply the model trained on the source task to target tasks without any adaptation. We can observe that the performance of the source model degrades drastically when the distributions of the target task are significantly different from the source task (*i.e.*, 15.4% drop from "dawn" to "'winter" and 10.9% drop from "dawn" to "winter night" with FCN-8s-ResNet101). On the other hand, ACE can effectively align feature distributions between the source task and target tasks, outperforming source only methods by clear margins. For example, ACE achieves a 3.4 and 9.6 (absolute percentage points) gain with FCN-8s-ResNet101 on HIGHWAY and NYC-LIKE CITY, respectively. In addition, we can see similar trends using both ResNet50-PSPNet and DeepLab V3+, confirming the fact that the framework is applicable to different top-performing networks for segmentation. Comparing across different networks, ResNet50-PSPNet offers the best mean mIoUs on both datasets after adaptation. Although DeepLab V3+ achieves the best results on the source task, its generalization ability is limited with more than 36.3% performance drop when applying to the "winter night" task. However, ACE can successfully bring back the performance with adaptation. Furthermore, we also observe that the performance on HIGHWAY is higher than that on NYC-LIKE CITY using different networks, which results from the fact the scenes are more cluttered with small objects like "traffic signs" in a city in contrast to highways. Figure 4 further visualizes the prediction maps generated by ACE and source only methods using ResNet50-PSPNet on HIGHWAY.

Method	Styles per task	HIGHWAY	NYC-LIKE CITY
Source	–	60.2	39.3
ACE	0	61.2	46.2
ACE	200	64.0	49.2
ACE	100	63.6	48.9

Table 2: **The performance of ACE depends on the number of exemplar style features stored in the memory unit.** The default number of features per task is 100, although we find that slight improvements can be made by increasing the number of stored vectors.

Effectiveness of style replay. We now investigate the performance of style replay using different numbers of feature vectors per task in the memory. Table 2 presents the results. The accuracy of ACE degrades by 2.4% and 2.9% on HIGHWAY and NYC-LIKE CITY respectively when no samples are used for replay, which confirms that style replay can indeed help revisiting previously learned knowledge to prevent forgetting. ACE without reply is still better than source only methods due to the fact the segmentation network is still being updated with inputs in different styles. When storing more exemplar feature vectors (*i.e.*, 200 per task) into the memory, ACE can be slightly improved by 0.4% and 0.3% on HIGHWAY and NYC-LIKE CITY, respectively. Here we simply use a random sampling approach to regenerate images in any of the historical styles, and we believe the sampling approach could be further improved with more advanced strategies [5].

Comparisons with prior art. We now compare with several recently proposed approaches based on FCN-8s-ResNet101: (1) SOURCE-REVERSE transfers testing images to the style of source images and then directly applies the segmentation model; (2) IADA aligns the feature distributions of the current task to those of a source task [63] in a sequential manner using adversarial loss functions [58] such that the feature distributions can no longer be differentiated by a trained critic; (3) ADDA-REPLAY stores previous samples and prediction scores and uses a matching loss to constrain the segmentation outputs from previous tasks to remain constant as adaptation progresses. The results are summarized in Table 3. We can see that ACE achieves the best results, outperforming other methods by clear margins, particularly on NYC-LIKE CITY where a 6.9% gain is achieved.

Method	HIGHWAY	NYC-LIKE CITY
Source	60.2	39.3
SOURCE-REVERSE	59.4	33.1
IADA [63]	60.7	40.4
ADDA-REPLAY [2]	61.9	42.0
ACE	63.6	48.9
ACE-ADDA	64.3	49.4

Table 3: **Comparisons with prior art**. We compare to two baseline methods, in addition to IADA [63] and ADDA-REPLAY [2].

Although SOURCE-REVERSE is a straightforward way to align feature distributions, the performance is worse than directly applying the source model. We suspect that this performance drop occurs because of small but systematic differences between the original source data on which the segmentation engine was trained, and the style transferred data on which no training ever occurs. In contrast, ACE trains the segmentation network on synthesized images, and constrains the segmentation output on generated images to be compatible with output on the original source image. In addition, IADA improves the source only model slightly by aligning feature distributions in a sequential manner, however, it relies on an adversarial loss function that is hard to optimize [1]. More importantly, while IADA proves to be successful for classification tasks, for tasks like segmentation where multiple classifiers are used for deep supervision [70, 30] at different distance scales, it is hard to know which feature maps to align to achieve the best performance. Further, we can also see that ADDA-REPLAY offers better results compared to IADA by using a memory to replay, however this requires storing all samples from previous tasks.

Note that ADDA [58] focuses on aligning distributions at the feature-level rather than the pixel-level, and this reduces low-level discrepancies in our approach. Yet, our approach is complimentary to approaches that explore feature-level alignment in the segmentation network at the cost of storing image samples for replay. When combining ADDA with ACE, 0.7% and 0.6% further improvements are achieved on both HIGHWAY and NYC-LIKE CITY.

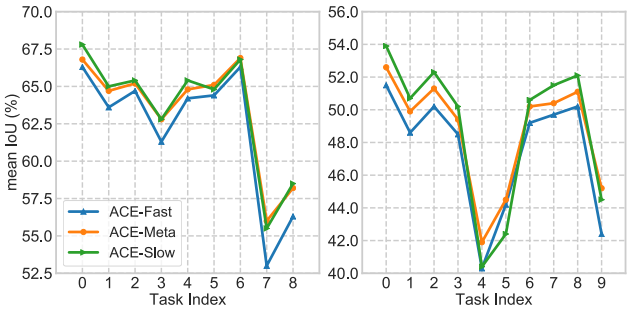

Figure 3: **Results of fast adaptations using ACE-Meta**. ACE-Slow: full batch training with SGD for 10K iterations. ACE-Fast: batch training with SGD using 600 iterations (the same number as ACE-Meta). Task indices correspond to task names in Table 1.

Fast adaptation with meta-updates. ACE achieves good results by batch training on each task using tens of thousands of SGD updates. We are also interested in how to adapt to the target task quickly by leveraging recent advances of meta-learning. We propose the meta-update method (ACE-Meta) which uses Reptile for learning meta-parameters, which are then fine-tuned to a specific task using only 600 iterations of SGD. We compare to ACE-Fast, which also uses 600 iterations per task, but without meta-learning, and also ACE-Slow, which uses full batch training with SGD for 10K iterations. The results are summarized in Figure 3. ACE-Meta achieves better performance compared to ACE-Fast, trained under the same settings, almost for all the target tasks on both HIGHWAY and NYC-LIKE

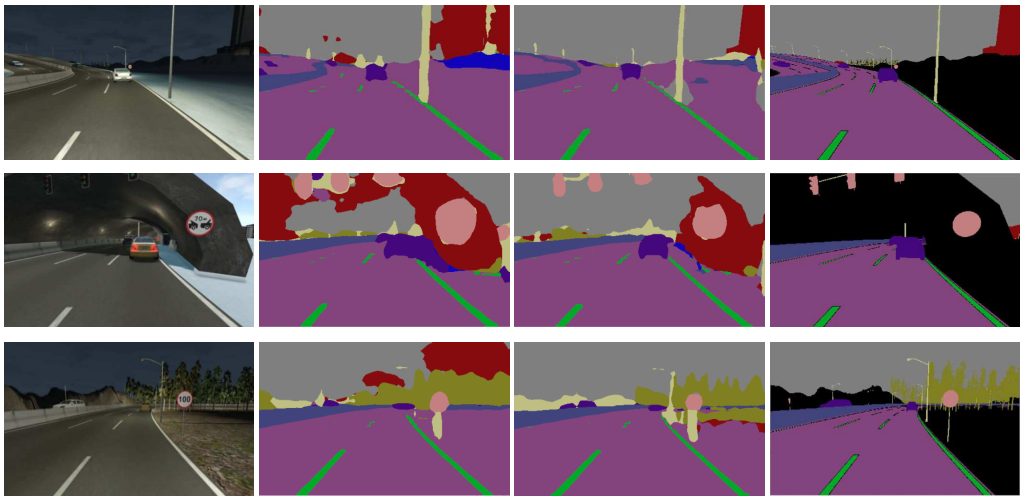

Figure 4: **Visualizations of prediction maps by different methods**. Prediction results from sampled images using ResNet50-PSPNet and the corresponding source only model. The color black indicates a class to be ignored during training.

CITY, and we observe clear gains when applying the model to "winter" and "winter night". Moreover, the results of ACE-Meta are on par with full batch training with SGD, demonstrating that meta-updates are able to learn the structures among different tasks.

Image generation with GANs. We compare images generated by ACE to MUNIT [20] in Figure 5. MUNIT learns to transfer the style of images from one domain to another by learning a shared space regularized by cycle consistency, and compared to CycleGAN [71], it is able synthesize a diverse set of results with a style encoder and a content encoder that disentangle the generation of style and content. Note that MUNIT also relies on AdaIN to control the style, but uses a GAN loss for generation. We can see that image generated with our approach preserves more detailed content (*e.g.*, facade of the building), and successfully transfers the snow to the walkway, while there are artifacts (*e.g.*, blurred regions) in the image generated with MUNIT.

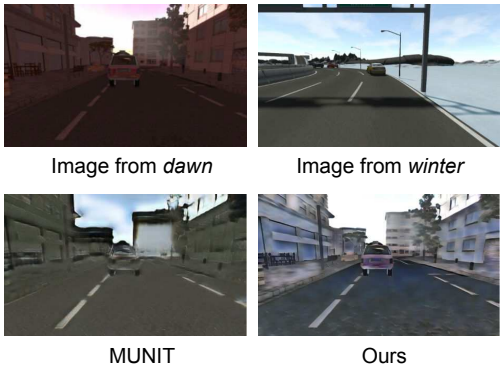

Figure 5: **Comparisons with different image synthesis methods**. Images generated with our method and comparisons with MUNIT by transferring the image from "dawn" to "winter".

5. Conclusion

We presented ACE, a framework that dynamically adapts a pre-trained model to a sequential stream of unlabeled tasks that suffer from domain shift. ACE leverages style replay to generalize well on new tasks without forgetting knowledge acquired in the past. In particular, given a new task, we introduced an image generator to align distributions at the pixel-level by synthesizing new images with the contents of the source task but in the style of the target task such that label maps from source images can be directly used for training the segmentation network. These generated images are used to optimize the segmentation network to adapt to new target distributions. To prevent forgetting, we also introduce a memory unit that stores the image statistics needed to produce different image styles, and replays these styles over time to prevent forgetting. We also study how meta-learning strategies can be used to accelerate the speed of adaptation. Extensive experiments are conducted on SYNTHIA and demonstrate that the proposed framework can effectively adapt to a sequence of tasks with shifting weather and lighting conditions. Future directions for research include how to handle distribution changes that involve significant geometry mismatch.

Acknowledgment Zuxuan Wu and Larry Davis are supported by Facebook and the Office of Naval Research under Grant N000141612713. Tom Goldstein is supported by DARPAs Lifelong Learning Machines and YFA programs, the NSF (DMS1912866), the AFOSR MURI program, and the Sloan Foundation.

References

[1] Martin Arjovsky and Léon Bottou. Towards principled methods for training generative adversarial networks. In *ICLR*, 2017. 7

[2] Andreea Bobu, Eric Tzeng, Judy Hoffman, and Trevor Darrell. Adapting to continuously shifting domains. In *ICLR Workshop*, 2018. 2, 7

[3] Konstantinos Bousmalis, Nathan Silberman, David Dohan, Dumitru Erhan, and Dilip Krishnan. Unsupervised pixel-level domain adaptation with generative adversarial networks. In *CVPR*, 2017. 2

[4] Konstantinos Bousmalis, George Trigeorgis, Nathan Silberman, Dilip Krishnan, and Dumitru Erhan. Domain separation networks. In *NIPS*, 2016. 2

[5] Francisco M. Castro, Manuel J. Marin-Jimenez, Nicolas Guil, Cordelia Schmid, and Karteek Alahari. End-to-end incremental learning. In *ECCV*, 2018. 3, 7

[6] Liang-Chieh Chen, Yukun Zhu, George Papandreou, Florian Schroff, and Hartwig Adam. Encoder-decoder with atrous separable convolution for semantic image segmentation. In *ECCV*, 2018. 6

[7] Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *ICML*, 2017. 3, 5

[8] Yaroslav Ganin and Victor S. Lempitsky. Unsupervised domain adaptation by backpropagation. In *ICML*, 2015. 2

[9] Yaroslav Ganin, Evgeniya Ustinova, Hana Ajakan, Pascal Germain, Hugo Larochelle, François Laviolette, Mario Marchand, and Victor Lempitsky. Domain-adversarial training of neural networks. *JMLR*, 2016. 2

[10] Leon A Gatys, Alexander S Ecker, and Matthias Bethge. Image style transfer using convolutional neural networks. In *CVPR*, 2016. 2, 4

[11] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *NIPS*, 2014. 2

[12] Dan Hendrycks and Thomas Dietterich. Benchmarking neural network robustness to common corruptions and perturbations. In *ILCR*, 2019. 1

[13] Geoffrey E. Hinton, Oriol Vinyals, and Jeffrey Dean. Distilling the knowledge in a neural network. *CoRR*, 2015. 4

[14] Judy Hoffman, Trevor Darrell, and Kate Saenko. Continuous manifold based adaptation for evolving visual domains. In *CVPR*, 2014. 2

[15] Judy Hoffman, Eric Tzeng, Taesung Park, Jun-Yan Zhu, Phillip Isola, Kate Saenko, Alexei A Efros, and Trevor Darrell. Cycada: Cycle-consistent adversarial domain adaptation. In *ICML*, 2018. 2

[16] Judy Hoffman, Dequan Wang, Fisher Yu, and Trevor Darrell. Fcns in the wild: Pixel-level adversarial and constraint-based adaptation. *CoRR*, 2016. 2, 5

[17] Weixiang Hong, Zhenzhen Wang, Ming Yang, and Junsong Yuan. Conditional generative adversarial network for structured domain adaptation. In *CVPR*, 2018. 2

[18] Haoshuo Huang, Qixing Huang, and Philipp Krahenbuhl. Domain transfer through deep activation matching. In *ECCV*, 2018. 2

[19] Xun Huang and Serge Belongie. Arbitrary style transfer in real-time with adaptive instance normalization. In *ICCV*, 2017. 2, 4

[20] Xun Huang, Ming-Yu Liu, Serge Belongie, and Jan Kautz. Multimodal unsupervised image-to-image translation. In *ECCV*, 2018. 2, 8

[21] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A. Efros. Image-to-image translation with conditional adversarial networks. In *CVPR*, 2017. 2

[22] Justin Johnson, Alexandre Alahi, and Li Fei-Fei. Perceptual losses for real-time style transfer and super-resolution. In *ECCV*, 2016. 2

[23] Tero Karras, Timo Aila, Samuli Laine, and Jaakko Lehtinen. Progressive growing of gans for improved quality, stability, and variation. In *ICLR*, 2018. 2

[24] James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, et al. Overcoming catastrophic forgetting in neural networks. *PNAS*, 2017. 3

[25] Christian Ledig, Lucas Theis, Ferenc Huszár, Jose Caballero, Andrew Cunningham, Alejandro Acosta, Andrew Aitken, Alykhan Tejani, Johannes Totz, Zehan Wang, et al. Photo-realistic single image super-resolution using a generative adversarial network. In *CVPR*, 2017. 2

[26] Yijun Li, Chen Fang, Jimei Yang, Zhaowen Wang, Xin Lu, and Ming-Hsuan Yang. Universal style transfer via feature transforms. In *NIPS*, 2017. 4

[27] Yanghao Li, Naiyan Wang, Jiaying Liu, and Xiaodi Hou. Demystifying neural style transfer. In *IJCAI*, 2018. 2, 4

[28] Ming-Yu Liu, Thomas Breuel, and Jan Kautz. Unsupervised image-to-image translation networks. In *NIPS*, 2017. 2

[29] Ming-Yu Liu Liu and Oncel Tuzel. Coupled generative adversarial networks. In *NIPS*, 2016. 2

[30] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In *CVPR*, 2015. 6, 7

[31] Mingsheng Long, Yue Cao, Jianmin Wang, and Michael I. Jordan. Learning transferable features with deep adaptation networks. In *ICML*, 2015. 2

[32] David Lopez-Paz et al. Gradient episodic memory for continual learning. In *NIPS*, 2017. 3

[33] Michael McCloskey and Neal J Cohen. Catastrophic interference in connectionist networks: The sequential learning problem. In *Psychology of learning and motivation*. 1989. 1

[34] Tsendsuren Munkhdalai and Hong Yu. Meta networks. In *ICML*, 2017. 3

[35] Alex Nichol, Joshua Achiam, and John Schulman. On first-order meta-learning algorithms. *CoRR*, 2018. 3, 5

[36] Augustus Odena, Christopher Olah, and Jonathon Shlens. Conditional image synthesis with auxiliary classifier gans. In *ICML*, 2017. 2

[37] Guim Perarnau, Joost van de Weijer, Bogdan Raducanu, and Jose M Álvarez. Invertible conditional gans for image editing. In *NIPS Workshop*, 2016. 2

[38] Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. In *ICLR*, 2016. 2

[39] Sachin Ravi and Hugo Larochelle. Optimization as a model for few-shot learning. In *ILCR*, 2017. 5

[40] Sylvestre-Alvise Rebuffi, Alexander Kolesnikov, Georg Sperl, and Christoph H Lampert. icarl: Incremental classifier and representation learning. In *CVPR*, 2017. 3

[41] Scott Reed, Zeynep Akata, Xinchen Yan, Lajanugen Logeswaran, Bernt Schiele, and Honglak Lee. Generative adversarial text to image synthesis. In *ICML*, 2016. 2

[42] Stephan R Richter, Vibhav Vineet, Stefan Roth, and Vladlen Koltun. Playing for data: Ground truth from computer games. In *ECCV*, 2016. 2, 5

[43] German Ros, Laura Sellart, Joanna Materzynska, David Vazquez, and Antonio M. Lopez. The synthia dataset: A large collection of synthetic images for semantic segmentation of urban scenes. In *CVPR*, 2016. 2

[44] Germán Ros, Simon Stent, Pablo F. Alcantarilla, and Tomoki Watanabe. Training constrained deconvolutional networks for road scene semantic segmentation. *CoRR*, 2016. 2, 5

[45] Kuniaki Saito, Kohei Watanabe, Yoshitaka Ushiku, and Tatsuya Harada. Maximum classifier discrepancy for unsupervised domain adaptation. In *CVPR*, 2018. 2

[46] Fatemeh Sadat Saleh, Mohammad Sadegh Aliakbarian, Mathieu Salzmann, Lars Petersson, and Jose M Alvarez. Effective use of synthetic data for urban scene semantic segmentation. In *ECCV*, 2018. 2

[47] Adam Santoro, Sergey Bartunov, Matthew Botvinick, Daan Wierstra, and Timothy Lillicrap. Meta-learning with memory-augmented neural networks. In *ICML*, 2016. 3

[48] Jürgen Schmidhuber, Jieyu Zhao, and Marco Wiering. Shifting inductive bias with success-story algorithm, adaptive levin search, and incremental self-improvement. *Machine Learning*, 1997. 3

[49] Wei Shen and Rujie Liu. Learning residual images for face attribute manipulation. In *CVPR*, 2017. 2

[50] Hanul Shin, Jung Kwon Lee, Jaehong Kim, and Jiwon Kim. Continual learning with deep generative replay. In *NIPS*, 2017. 3

[51] Konstantin Shmelkov, Cordelia Schmid, and Karteek Alahari. Incremental learning of object detectors without catastrophic forgetting. In *ICCV*, 2017. 3

[52] Ashish Shrivastava, Tomas Pfister, Oncel Tuzel, Joshua Susskind, Wenda Wang, and Russell Webb. Learning from simulated and unsupervised images through adversarial training. In *CVPR*, 2017. 2

[53] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. In *ICLR*, 2015. 4

[54] Baochen Sun and Kate Saenko. Deep CORAL - correlation alignment for deep domain adaptation. In *ECCV*, 2016. 2

[55] Yaniv Taigman, Adam Polyak, and Lior Wolf. Unsupervised cross-domain image generation. In *ICLR*, 2017. 2

[56] Sebastian Thrun and Lorien Pratt. Learning to learn: Introduction and overview. In *Learning to learn*. 1996. 3

[57] Eric Tzeng, Judy Hoffman, Trevor Darrell, and Kate Saenko. Simultaneous deep transfer across domains and tasks. In *ICCV*, 2015. 2

[58] Eric Tzeng, Judy Hoffman, Kate Saenko, and Trevor Darrell. Adversarial discriminative domain adaptation. In *CVPR*, 2017. 2, 7

[59] Dmitry Ulyanov, Vadim Lebedev, Andrea Vedaldi, and Victor S Lempitsky. Texture networks: Feed-forward synthesis of textures and stylized images. In *ICML*, 2016. 2, 4

[60] Dmitry Ulyanov, Andrea Vedaldi, and Victor S. Lempitsky. Instance normalization: The missing ingredient for fast stylization. *CoRR*, 2016. 2

[61] Oriol Vinyals, Charles Blundell, Timothy Lillicrap, Daan Wierstra, et al. Matching networks for one shot learning. In *NIPS*, 2016. 3

[62] Zuxuan Wu, Xintong Han, Yen-Liang Lin, Mustafa Gokhan Uzunbas, Tom Goldstein, Ser Nam Lim, and Larry S Davis. Dcan: Dual channel-wise alignment networks for unsupervised scene adaptation. In *ECCV*, 2018. 2, 4

[63] Markus Wulfmeier, Alex Bewley, and Ingmar Posner. Incremental adversarial domain adaptation for continually changing environments. In *ICRA*, 2018. 2, 7

[64] Ju Xu and Zhanxing Zhu. Reinforced continual learning. In *NIPS*, 2018. 3

[65] Donggeun Yoo, Namil Kim, Sunggyun Park, Anthony S. Paek, and In-So Kweon. Pixel-level domain transfer. In *ECCV*, 2016. 2

[66] Jaehong Yoon, Eunho Yang, Jeongtae Lee, and Sung Ju Hwang. Lifelong learning with dynamically expandable networks. In *ICLR*, 2018. 3

[67] Friedemann Zenke, Ben Poole, and Surya Ganguli. Continual learning through synaptic intelligence. In *ICML*, 2017. 3

[68] Yang Zhang, Philip David, and Boqing Gong. Curriculum domain adaptation for semantic segmentation of urban scenes. In *ICCV*, 2017. 2, 5

[69] Yiheng Zhang, Zhaofan Qiu, Ting Yao, Dong Liu, and Tao Mei. Fully convolutional adaptation networks for semantic segmentation. In *CVPR*, 2018. 2

[70] Hengshuang Zhao, Jianping Shi, Xiaojuan Qi, Xiaogang Wang, and Jiaya Jia. Pyramid scene parsing network. In *CVPR*, 2017. 6, 7

[71] Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A. Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. In *ICCV*, 2017. 2, 8