Stochastic Classifiers for Unsupervised Domain Adaptation

Zhihe Lu[1,2] Yongxin Yang[1,2] Xiatian Zhu[1] Cong Liu[2] Yi-Zhe Song[1,2] Tao Xiang[1,2]
[1]CVSSP, University of Surrey [2]iFlyTek-Surrey Joint Research Centre on Artificial Intelligence
{zhihe.lu, yongxin.yang, xiatian.zhu, y.song, t.xiang}@surrey.ac.uk, congliu2@iflytek.com

Abstract

A common strategy adopted by existing state-of-the-art unsupervised domain adaptation (UDA) methods is to employ two classifiers to identify the misaligned local regions between a source and target domain. Following the 'wisdom of the crowd' principle, one has to ask: why stop at two? Indeed, we find that using more classifiers leads to better performance, but also introduces more model parameters, therefore risking overfitting. In this paper, we introduce a novel method called STochastic clAssifieRs *(STAR) for addressing this problem. Instead of representing one classifier as a weight vector, STAR models it as a Gaussian distribution with its variance representing the inter-classifier discrepancy. With STAR, we can now sample an arbitrary number of classifiers from the distribution, whilst keeping the model size the same as having two classifiers. Extensive experiments demonstrate that a variety of existing UDA methods can greatly benefit from STAR and achieve the state-of-the-art performance on both image classification and semantic segmentation tasks.*

1. Introduction

Remarkable advances on image classification accuracy have been realised in the *supervised learning* paradigm [14, 20, 48, 52]. This success is based on two assumptions: there are hundreds/thousands of labelled training images per class available for model training, and the training and test data are drawn from the same domain, and with similar distributions. However, collecting such a big training set for every target domain is prohibitively expensive and time-consuming in large scale real-world applications. An intuitive solution is to transfer knowledge from an available richly-labelled domain (*i.e*., source domain) to a target domain without labelled training data but containing the same set of classes. Often, the data distributions of the source domain and target domain are different significantly, which renders the model trained/specialised on the source domain *not* directly applicable to the target domain. Unsupervised Domain Adaptation (UDA) provides an effective approach to solving this problem [10, 29].

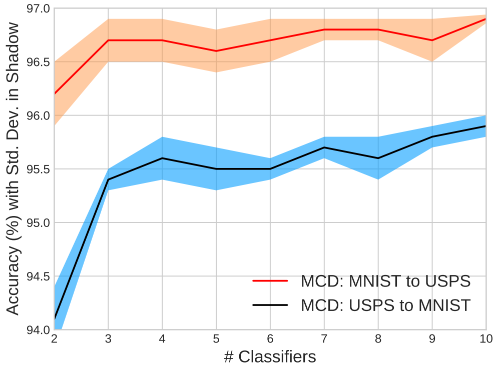


Figure 1: The test accuracy vs. classifier number using the Maximum Classifier Discrepancy (MCD) model [45] on two digit classification UDA tasks. The solid line is the average accuracy over 5 trials, whilst the standard deviation is represented by shadows. The best results are clearly not obtained with two classifiers.

There have been a large spectrum of UDA methods developed in the literature [36, 56]. From a distribution alignment perspective, existing UDA methods can be divided generally into two groups: (i) *Global Alignment* (GA) methods [10, 28, 15] and (ii) *Local Alignment* (LA) methods [45, 22, 31]. Considering *holistically* the per-domain data distribution as a whole, GA methods tend to overlook the *local* class decision boundary information during adaptation, thereby resulting in sub-optimal performance on the target domain. This limitation can be addressed by the recent LA methods which take into account class-level cross-domain distribution alignment. Concretely, LA methods first leverage the disagreement of a small number (typically two) of classifiers, to identify the misalignment areas between source and target distributions in a joint feature embedding space. Subsequently, a feature extraction model is trained in a way such that the discovered misalignment can be minimised in the embedding space. The two operations are repeated alternately until convergence.

Whilst the LA methods [45, 22, 31] yield the state-of-the-art UDA performance on various benchmarks with two classifiers, a fundamental yet largely ignored question is:

Task	USPS ↕ MNIST	SVHN ↓ MNIST	SYNTH ↓ GTRSB
# Para. of G	0.04 M	25.5 M	1.1 M
# Para. of C	0.18 M	12.6 M	6.6 M
# Total Para.	0.22 M	38.1 M	7.7 M

Table 1: The number of parameters of the feature extractor G and the two classifiers C in MCD [45] on different tasks. Obviously, the classifier parameters take the majority part in most cases. This means adding more classifiers will increase the model size significantly.

what is the optimal number of classifiers for the LA model design? In particular, why stop at two when the 'wisdom of the crowd' principle suggests that the more the merit. To answer this, we start by experimenting a varying number of classifiers in MCD (Maximum Classifier Discrepancy) [45] on two digit classification tasks. As shown in Figure 1, the optimal classifier number is largely *task-specific*, and using *more classifiers* results in better model performances in general. This analysis hence suggests that LA methods should use *more than two* classifiers. A plausible rationale behind is that using more classifiers can identify and explore more comprehensively misaligned local regions in the Dempster-Shafer theory of evidence [41]. However, it is nontrivial to design a principled method for estimating the optimal classifier number. Besides, simply adding many classifiers to existing methods will not only lead to higher computation complexity (a quadratic cost in the number of classifier for computing the pairwise classifier discrepancy), but also significantly increase the model parameter number (cf. Table 1) and suffer from a higher risk of overfitting.

To overcome the above problems, in this work we introduce *STochastic clAssifieRs* (STAR) to integrate an approximately *infinite* number of classifiers into existing LA methods *without* adding more parameters nor extra computational overhead. With STAR, the classifiers are represented by a weight distribution, rather than specific weight points as in the conventional LA methods. Concretely, the classifiers are modelled with a Gaussian distribution that needs to be optimised in training. That is, we consider a classifier weight vector as a random variable. The mean of the distribution serves as the final classifier weight whilst the variance represents the discrepancy (*i.e.*, disagreement) degree of different classifiers. In each training iteration, a small number of (*e.g.*, two) *different* new classifiers are sampled randomly from the current distribution estimate, finally leading to a large number of classifiers being sampled over the entire training process with many iterations. Consequently, the UDA model is allowed to be trained with far more different classifiers than before. Importantly, this is achieved without extra need for tuning the classifier number whilst avoiding the negative effects of using many specific classifiers and increasing the model size.

However, the inference and training of a deep CNN that models the classifiers as a weight distribution is nontrivial. This is because the random sampling of classifiers at each iteration prevents the conventional end-to-end training. To solve this issue, a reparameterisation trick is introduced to enable the STAR model to be trainable with any off-the-shelf optimiser. This enables the direct use of existing LA model's loss functions, making our STAR generally applicable and usable in a plug-in manner.

We make the following contributions: **(1)** We identify the significance of using many classifiers in the state-of-the-art local alignment UDA methods to model performance. To the best of our knowledge, this is the first attempt to investigate this issue in UDA. **(2)** We formulate a novel solution for addressing the classifier scalability issue for UDA by introducing *STochastic clAssifieRs* (STAR) that enables existing LA methods to leverage an approximately infinite number of classifiers for improved local domain misalignment identification. STAR is a generically stochastic UDA framework which can benefit any previous methods using multi-classifiers. This is also the very first work that introduces the stochastic deep learning concept into the UDA problem, to our knowledge, expanding the application scope of stochastic deep learning. **(3)** With extensive evaluations on both image classification and semantic segmentation tasks, we show that a variety of existing LA methods benefit from the proposed STAR, often resulting in large improvement and state-of-the-art performance.

2. Related Work

Distribution Alignment Distribution alignment between different domains is a common way to alleviate domain shifts in unsupervised domain adaptation (UDA) tasks. Previous methods based on that can be divided into *global alignment* (GA) and *local alignment* (LA). For GA, many metrics, including Maximum Mean Discrepancy (MMD) [28, 29], Central Moment Discrepancy (CMD) [60] and Wasserstein distance [47], have been proposed or utilised. Since these methods neglected the local class-level alignment, they may reach a sub-optimal solution. To solve this problem, most recent UDA methods are based on some forms of LA. In particular, [15] introduced a cycle-consistency loss, matching a pixel-level distribution. Similarly, MCD [45] adapted the target features by aligning the outputs of diverse classifiers and CLAN [31] designed a category-level adversarial loss for semantic segmentation. Based on MCD [45], [22] designed a novel discrepancy loss to diversify classifiers. In this work, we focus on improving the LA methods that rely on multiple classifiers to identify local misalignment regions in a feature embedding space.

Adversarial Training Regardless of whether the distribution alignment is done globally or locally, one common way to align source and target domain data distribution is via adversarial training [12]. These methods can be roughly separated into three groups depending on which level the adversarial training is introduced: feature-level [11, 45], pixel-level [2, 27] and output-level [54]. To make features discriminative for the classification task on the source domain and indiscriminate concerning domain shifts, [11] proposed a new gradient reversal layer for the domain adversarial training. In the pixel-level methods, [2] designed a style transferring method based on GAN [12] and used the transferred results on the target domain for UDA scenarios. [54] considered semantic segmentation has structured outputs, so they conducted the adversarial training on the output level. In contrast, [45] proposed a novel within-network adversarial training strategy whereby a feature generator competes with two task-specific classifiers. Based on adversarial training, some other strategies, *e.g.*, dropout [44] and domain-specific batch normalisation (DSBN) [3], have been proposed to impose on previous methods. In this paper, we follow the adversarial training pipeline, but it is worth noting that our method can also be applied to non-adversarial based methods such as [43, 61] as long as they require multiple classifiers.

Stochastic Neural Networks Conventionally, neural network models are deterministic, *i.e.*, their parameters/weights are point-estimate. Thus, they cannot model the uncertainty and usually produce the predictions in an overly confident way [1]. In contrast, stochastic neural networks, e.g., Bayesian Neural Networks (BNNs) [33, 9], can deliver the intermediate products and/or final predictions in the form of distribution, which can lead to richer representations. Recently, stochastic neural networks have been applied to several computer vision problems. For example, [50] proposed an uncertainty aware multi-modal BNNs for activity recognition and [59] modelled the feature uncertainty using distributions in person re-identification. [19] utilised a Gaussian distribution to model the latent variables of input images as a means for data augmentation. Differently, in this work, stochastic models is used for UDA for the first time.

3. Methodology

3.1. Problem Setting

We study the problem of unsupervised domain adaptation (UDA) for classification and segmentation. We have access to source domain data X_S along with their labels Y_S. Meanwhile, we have target domain data X_T which is unlabelled but shares the same label space with X_S. The objective is to train a classifier using $\{X_S, Y_S\}$ and X_T that generalises to the target domain.

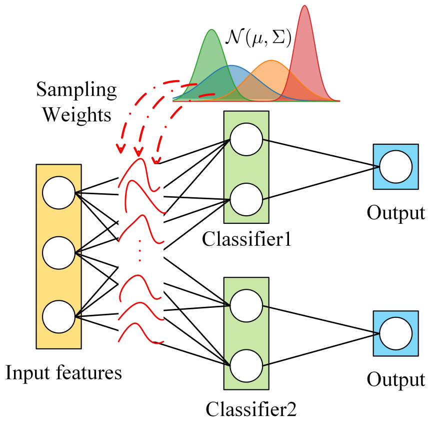


Figure 2: The architecture of STochastic clAssifieRs (STAR). At each training iteration, the classifier weights are sampled randomly from the distribution whilst the parameters of the distribution are simultaneously optimised.

3.2. Diverse Classifiers

Recently, several studies utilised the concept of *two diverse classifiers* [53] in ensemble learning for unsupervised domain adaptation. For example, MCD [45] proposed to use two classifiers for class-level local alignment with a focus on the classification tasks. CLAN [31] instead focuses on the semantic segmentation problem. It trained two classifiers, measured their outputs difference, and weighted the pixel level adversarial loss according to that difference. Note that the diversity of the two classifiers is realised by different mechanisms. MCD maximises the classifier discrepancy on target domain in ℓ_1 norm, whilst CLAN minimises the cosine similarity of the classifier parameters.

3.3. Stochastic Classifiers

Empirically we found that increasing the number of classifiers usually improves the model performance (cf. Figure 1). However, this comes with a number of limitations as discussed earlier. The core idea of our method is to model a distribution of classifiers. Then the classifiers used for local discrepancy discovery are simply random samples of the distribution. Modelling classifier distribution gives access to the infinite number of classifiers, since we can sample an arbitrary number of classifiers. Importantly, this decouples the number of classifiers and model parameters, enabling us to obtain many classifiers whilst keep the overall model size unchanged thus avoiding the overfitting risk. As the classifiers are *independently* sampled from the in-training distribution at each iteration, we finally obtain a sufficiently large number of classifiers. This further eliminates the need to tune the number of classifiers as an extra hyper-parameter. Due to the nature of randomness that allows the classification behaviour to be analysed *statistically*, we name the proposed method as ***STochastic clAssifieRs*** (**STAR**) as illustrated in Figure 2.

More specifically, we build a multivariate Gaussian distribution $\mathcal{N}(\mu, \Sigma)$, parametrised by a mean vector u and a diagonal covariance matrix Σ. Whenever we need a number of classifiers, we can sample from $\mathcal{N}(\mu, \Sigma)$, and the relevant loss will be back-propagated to the learnable parameters u and Σ. The choice of multivariate Gaussian for the classifier distribution is twofold: (i) it is reparametrisable [19], which is crucial for backpropagation; (ii) with diagonal Σ, the number of trainable parameters is exactly the same as the two classifier case.

3.4. Instantiation

Since we can sample any number of classifiers and backpropagate the errors back to the distribution parameters, *i.e.*, μ and Σ, the proposed method can be brought into any problem-specific solutions with a multi-classifier ingredient. Here we instantiate two types of application – image classification (based on MCD [45]) and semantic segmentation (based on CLAN [31]).

3.4.1 Image Classification

MCD [45] consists of three modules: (i) the feature extraction network $g_\theta(\cdot)$, (ii) the first classifier $f_{\phi_1}(\cdot)$, (iii) the second classifier $f_{\phi_2}(\cdot)$. The objective for the feature extraction network is to cooperate with either of the classifiers for recognition on source domain, and to *minimise* the prediction discrepancy of two classifiers on target domain. Meanwhile, the objective for having two classifiers is to recognise the object accurately for the source domain whilst *maximising* the discrepancy for the target domain.

More specifically, MCD optimisation is scheduled by alternating,

Step A $\min\limits_{\theta,\phi_1,\phi_2} \ell(f_{\phi_1}(g_\theta(x_S)), y_S) + \ell(f_{\phi_2}(g_\theta(x_S)), y_S)$

Step B $\max\limits_{\phi_1,\phi_2} \|f_{\phi_1}(g_\theta(x_T)) - f_{\phi_2}(g_\theta(x_T))\|_1$

Step C $\min\limits_{\theta} \|f_{\phi_1}(g_\theta(x_T)) - f_{\phi_2}(g_\theta(x_T))\|_1$

where $\{x_S, y_S\}$ is a mini-batch from source domain, x_T is a mini-batch from target domain, and $\ell(\cdot,\cdot)$ is the cross-entropy loss. Note that, the source domain cross-entropy loss of both classifiers can be further added to **Step B** for stabilising the optimisation process.

To equip MCD with our stochastic classifiers, we can simply swap $\{\phi_1, \phi_2\}$ with $\{\tilde{\phi}_1, \tilde{\phi}_2\}$, where $\tilde{\phi}_1$ and $\tilde{\phi}_2$ are two independent samples drawn from $\mathcal{N}(\mu, \Sigma)$.

The sampling process is usually non-differentiable, thus we adapt the *reparametrisation trick*, *i.e.*, $\tilde{\phi}_1 = \mu + \sigma \odot \epsilon_1$ and $\tilde{\phi}_2 = \mu + \sigma \odot \epsilon_2$. Here ϵ_1 and ϵ_2 are two independent samples drawn from a standard Gaussian. $\odot$ denotes element-wise product and σ is the diagonal of Σ.

3.4.2 Semantic Segmentation

CLAN [31] is based on adversarial domain adaptation [10]. It consists of four modules: (i) the feature extraction network $g_\theta(\cdot)$, (ii) the first classifier $f_{\phi_1}(\cdot)$, (iii) the second classifier $f_{\phi_2}(\cdot)$, (iv) the domain classifier $h_\psi(\cdot)$. The core is a binary classification loss with source domain instances being positive and target domain instances being negative, *i.e.*,

$$\begin{aligned}\ell^{(\mathrm{A})}_{\theta,\phi_1,\phi_2,\psi}(x_S, x_T) = - & \log(h_\psi(f_{\phi_1}(g_\theta(x_S)))) \\ - & \log(h_\psi(f_{\phi_2}(g_\theta(x_S)))) \\ & -\rho \log(1 - h_\psi(f_{\phi_1}(g_\theta(x_T)))) \\ & -\rho \log(1 - h_\psi(f_{\phi_2}(g_\theta(x_T)))) \quad (1)\end{aligned}$$

and the min-max optimisation is built as,

$$\min_{\psi} \max_{\theta,\phi_1,\phi_2} \ \ell^{(\mathrm{A})}_{\theta,\phi_1,\phi_2,\psi}(x_S, x_T) \qquad (2)$$

CLAN has a weighting factor for the last two terms of Eq. 1, and the factor is computed by the cosine distance of two classifiers' predictions, *i.e.*, $\rho = 1 - \frac{p_1^T p_2}{\|p_1\|\|p_2\|}$ where $p_1 = f_{\phi_1}(g_\theta(x_T))$ and $p_2 = f_{\phi_2}(g_\theta(x_T))$. Intuitively, this downplays the importance of instances that are already well-aligned.

To enforce divergence of two classifiers, CLAN uses a loss based on the cosine similarity of their parameters, *i.e.*,

$$\ell^{(\mathrm{W})}_{\phi_1,\phi_2} = \frac{\phi_1^T \phi_2}{\|\phi_1\|\|\phi_2\|} \qquad (3)$$

The full objective of CLAN is optimised by alternating two steps,

Step 1 $\min\limits_{\theta,\phi_1,\phi_2} \ell(\hat{y}^{(1)}_S, y_S) + \ell(\hat{y}^{(2)}_S, y_S) + \ell^{(\mathrm{W})} - \ell^{(\mathrm{A})}(x_T)$

Step 2 $\min\limits_{\psi} \ell^{(\mathrm{A})}(x_S, x_T)$

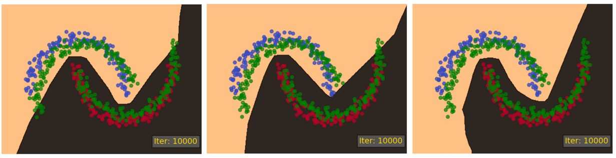
where $\ell(\cdot,\cdot)$ is the segmentation loss (multi-class cross-entropy loss at pixel level), $\hat{y}^{(1)}_S = f_{\phi_1}(g_\theta(x_S))$, $\hat{y}^{(2)}_S = f_{\phi_2}(g_\theta(x_S))$ and $\ell^{(\mathrm{A})}(x_T)$ are the last two terms of Eq. 1.

Similarly, we can equip CLAN with our stochastic classifiers by setting $\phi_1 \leftarrow \mu + \sigma \odot \epsilon_1$ and $\phi_2 \leftarrow \mu + \sigma \odot \epsilon_2$ where ϵ_1 and ϵ_2 are two independent samples drawn from standard Gaussian.

In the same manner as the above examples, other existing methods with two or more classifiers such as co-training [43] and tri-training [61] can be reformulated by our STAR method.

3.5. Further Analysis

Toy Problem We run a toy experiment on the well-known two-moon dataset. For source domain data, we generate an



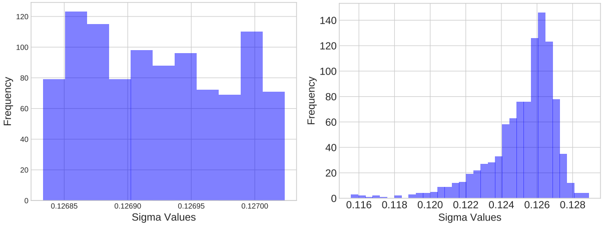
(a) Source only (b) MCD (c) Ours

Figure 3: Toy experiments on the two-moon 2D dataset. The blue and red points (the source domain) belong to class 0, 1, respectively, whilst the green points are the target domain data. The decision boundary divides different classes in black and flesh tone colours respectively. (a) Trained with source only model. (b) MCD [45]. (c) Our STAR.

upper moon and a lower moon, representing two different classes. By rotating the source domain data, we get target domain data, as shown in Figure 3. There are 300 samples per class for both source and target domains. In all experiments, we use a three-layer MLP for the feature generator and two separate three-layer MLPs for classifiers. For our method, we replace the final layer in the classifier with a stochastic layer. We train all the methods for $10,000$ iterations to guarantee convergence. We show the decision boundaries from source only model (Figure 3(a)), MCD (Figure 3(b)) (the final decision boundary is the mean of two decision boundaries of two classifiers) and our method (Figure 3(c)). We can tell that our method has the best decision boundary that classifies all the target samples correctly, whilst MCD failed to locate and align some misaligned data points at the right end of the lower moon for having access to only two classifiers.

How STAR Works The improved performance is credited to the variance of distribution Σ. As we can see from Figure 4, the initial values of Σ are uniformly distributed, but they become more patterned after training. This explains how our method works: (i) the classifier distribution tends to have larger variances for the misaligned features (data points); (ii) the feature extractor will counter (i) by alleviating the misalignment identified by those larger variances; (iii) the classifier will push variance larger for any existing misaligned features. Finally, this process will arrive at a balance point: the features are aligned as much as possible, and the large variances remain as there is no further motivation to reduce them. This results in what we have observed in Figure 4: the proportion of large variances increases from epoch 0 to epoch 300.

So why the vanilla MCD is less effective? Indeed, two classifiers can close the loop of identifying-aligning features, but more classifiers leads to better effectiveness. This is intuitive because it is fairly easy to make two classifies agree with each, and it becomes harder when more classifiers join in, as they may focus on different features and disagree because of those features.



(a) Epoch 0. (b) Epoch 300.

Figure 4: The distribution of the flattened Σ values for (a) initialisation (b) after convergence of STAR on MNIST to USPS task.

Method	SVHN ↓ MNIST	SYNSIG ↓ GTSRB	MNIST ↓ USPS	USPS ↓ MNIST
Source only	67.1	85.1	79.4	63.4
DANN [11]	84.2	-	90.4	94.7
ADDA [55]	76.0±1.8	-	-	90.1±0.8
CoGAN [27]	-	-	-	89.1 ± 0.8
PixDA [2]	-	-	95.9	-
ASSC [13]	95.7±1.5	82.8±1.3	-	-
UNIT [26]	90.5	-	96.0	93.6
CyCADA [15]	90.4±0.4	-	95.6 ± 0.2	96.5±0.1
GTA [46]	92.4±0.9	-	95.3±0.7	90.8±1.3
DeepJDOT [7]	96.7	-	95.7	96.4
SimNet [38]	-	-	96.4	95.6
GICT [39]	98.7	-	96.2	96.6
MCD [45]	96.2±0.4	94.4±0.3	96.5±0.3	94.1±0.3
STAR	**98.8**±0.05	**95.8**±0.2	**97.8**±0.1	**97.7**±0.05

Table 2: The digit and traffic sign classification performance. We reported the mean and the standard deviation of the accuracy obtained over 5 trials.

Testing-stage Prediction Previous methods with multiple classifiers usually resort to the feature/score fusion or voting by majority [17, 51] for a final decision. With the classifiers predicting in diverse views, a more robust decision can be made. Since we use stochastic classifiers, theoretically, we can fuse an arbitrary number of predictions. However, a simple yet effective way is to use the mean value μ for the final prediction, and we find this works well empirically.

4. Experiment

4.1. Image Classification

4.1.1 Digit and Sign Classification

Datasets In this experiment, we used three digit datasets (MNIST [21], Street View House Numbers (SVHN) [34] and USPS [16]), and two sign datasets (Synthetic Traffic Sign (SYNSIGNS) [32] and the German Traffic Signs Recognition Benchmark (GTSRB) [49]). In terms of domain characteristics, MNIST contains grayscale digit images with the clean background; SVHN [34] consists of cropped coloured digits from real scenes with extremely blurred appearance; USPS provides grayscale handwrit-

Method	plane	bcycl	bus	car	horse	knife	mcycl	person	plant	sktbrd	train	truck	**mean**
Source Only	55.1	53.3	61.9	59.1	80.6	17.9	79.7	31.2	81.0	26.5	73.5	8.5	52.4
MMD [28]	87.1	63.0	76.5	42.0	90.3	42.9	85.9	53.1	49.7	36.3	85.8	20.7	61.1
DANN [10]	81.9	77.7	82.8	44.3	81.2	29.5	65.1	28.6	51.9	54.6	82.8	7.8	57.4
JAN [30]	75.7	18.7	82.3	**86.3**	70.2	56.9	80.5	53.8	92.5	32.2	84.5	54.5	65.7
ADR [44]	87.8	79.5	83.7	65.3	92.3	61.8	88.9	73.2	87.8	60.0	85.5	32.3	74.8
DEV [58]	81.83	53.48	82.95	71.62	89.16	72.03	89.36	75.73	**97.02**	55.48	71.19	29.17	72.42
GICT [39]	87.6	60.6	81.6	72.1	87.8	62.9	89.7	68.5	88.8	76.1	83.2	20.0	73.1
LPJT [24]	93.0	80.3	66.5	56.3	**95.8**	70.3	74.2	**83.8**	91.7	40.0	78.7	**57.6**	74.0
BSP (CDAN) [5]	92.4	61.0	81.0	57.5	89.0	80.6	90.1	77.0	84.2	77.9	82.1	38.4	75.9
DSBN (MSTN) [3]	94.7	**86.7**	76.0	72.0	95.2	75.1	87.9	81.3	91.1	68.9	88.3	45.5	80.2
SAFN [57]	93.6	61.3	84.1	70.6	94.1	79.0	**91.8**	79.6	89.9	55.6	**89.0**	24.4	76.1
TPN [35]	93.7	85.1	69.2	81.6	93.5	61.9	89.3	81.4	93.5	81.6	84.5	49.9	80.4
DTA [23]	93.7	82.2	**85.6**	83.8	93.0	81.0	90.7	82.1	95.1	78.1	86.4	32.1	81.5
MCD [45]	87.0	60.9	83.7	64.0	88.9	79.6	84.7	76.9	88.6	40.3	83.0	25.8	71.9
STAR	**95.0**	84.0	84.6	73.0	91.6	**91.8**	85.9	78.4	94.4	**84.7**	87.0	42.2	**82.7**

Table 3: The object classification performance on the VisDA 2017 [37] benchmark. All the methods use the ResNet101 model [14] pretrained on ImageNet as the backbone model.

ten digit images with unconstrained writing styles. Whilst sharing the same 10 (0∼9) digit classes, the three datasets present significantly different data distributions, therefore suitable for UDA evaluation. For sign datasets, SYNSIGNS gives synthesised traffic signs which appear dramatically different from the real-world images of GTSRB. So the two form a good UDA task with 43 object classes. For UDA test, we adopted four commonly used cross-dataset transfer settings with the standard data split: SVHN ⇒ MNIST, USPS ⇒ MNIST, MNIST ⇒ USPS, SYNSIGN ⇒ GTSRB [45].

Model instantialisation We adopted MCD [45] as the UDA framework in this experiment[1]. For a fair comparison, we used the same network designs for feature extractor and classifier as [45]. Note that we only replace the last FC layer of classifier with a STAR layer whilst sharing other layers.

Training details We used Adam [18] as the optimiser with the learning rate 2×10^{-4} and the batch size 128. We trained 200 epochs for SVHN ⇒ MNIST and SYNSIGN ⇒ GTSRB, and 300 epochs for USPS ⇒ MNIST and MNIST ⇒ USPS (due to less training data). We followed the same hyper-parameter setting as MCD [45] without extra tuning.

Results We evaluated the performance of STAR in comparison to a wide range of existing state-of-the-art methods on the four test settings in Table 2. We made the following observations: **(1)** Directly applying the model trained on the source domain data yields weak performance, due to the data distribution gap between source and target domains. **(2)** Compared with the backbone method MCD [45], our STAR improves consistently the recognition accuracy by 2.3% on average over all UDA tasks. This suggests the dataset-agnostic efficacy of STAR, validating our idea of exploiting stochastic classifiers enhancing the ability of LA-based UDA methods to identifying local misalignment. **(3)** With such improvement, STAR outperforms all compared methods, often by a large margin with low accuracy variances. This low variance also suggests that modelling classifier distribution makes STAR less sensitive to the random initialisation in different trials.

4.1.2 Object Classification

Datasets We evaluated a more challenging object classification task which transfers the knowledge of synthetic images in VisDA [37] to classify real images in COCO [25]. VisDA contains 152,397 synthetic images from 12 classes. The target test data is a set of 55,388 COCO validation images from the same classes.

Model instantialisation For a fair comparison with existing methods, we used the same backbone ResNet101 [14] pretrained on ImageNet [8], as in [45, 23]. We selected the MCD [45] as the UDA pipeline. That is, we discarded the last FC layer in ResNet101 and used the rest as the feature generator. We then deployed our stochastic classifiers with 3 FC layers.

Training details We used the input image size of 224 × 224. We adopted the SGD optimiser with the batch size 32, the learning rate 1.0×10^{-3} for both the feature extractor and classifiers.

Results We compared the object classification accuracy of STAR on VisDA with a wide variety of state-of-the-art UDA methods in Table 3. We have similar observations as digit/sign classification above, *e.g.*, our STAR achieves again the best overall performance. A notable difference is that STAR yields a remarkable 10.8% improvement over the baseline MCD on this task, which is significantly larger than that achieved on the simpler digit/sign recognition tasks (averaged 2.3%). This is encouraging as it suggests that our method is better at solving more challenging recognition tasks.

[1] We did not experiment with SWD [22] since we cannot reproduce the reported results even with the help of the authors.

Method	road	side.	buil.	wall	fence	pole	light	sign	vege.	terr.	sky	pers.	rider	car	truck	bus	train	motor	bike	**mIoU**
Source Only	75.8	16.8	77.2	12.5	21.0	25.5	30.1	20.1	81.3	24.6	70.3	53.8	26.4	49.9	17.2	25.9	6.5	25.3	**36.0**	36.6
ASN (feature) [54]	83.7	27.6	75.5	20.3	19.9	27.4	28.3	**27.4**	79.0	28.4	70.1	55.1	20.2	72.9	22.5	35.7	**8.3**	20.6	23.0	39.3
ASN (single-level) [54]	86.5	25.9	79.8	22.1	20.0	23.6	33.1	21.8	81.8	25.9	75.9	57.3	26.2	76.3	29.8	32.1	7.2	29.5	32.5	41.4
ASN (multi-level) [54]	86.5	36.0	79.9	23.4	23.3	23.9	35.2	14.8	83.4	33.3	75.6	58.5	27.6	73.7	32.5	35.4	3.9	30.1	28.1	42.4
GICT [39]	**88.6**	**41.3**	76.4	23.3	**26.1**	24.3	32.8	23.1	82.3	**37.4**	73.3	**62.2**	24.8	73.3	29.6	33.9	4.6	**33.4**	24.3	42.8
CLAN [31]	87.0	27.1	79.6	**27.3**	23.3	**28.3**	**35.5**	24.2	**83.6**	27.4	74.2	58.6	**28.0**	76.2	33.1	36.7	6.7	31.9	31.4	43.2
CLAN†	87.9	28.6	79.9	24.0	24.0	24.9	33.3	19.4	83.4	31.9	76.1	58.3	27.5	82.8	**35.4**	**42.6**	0.7	27.1	27.2	42.9
STAR	88.4	27.9	**80.8**	**27.3**	25.6	26.9	31.6	20.8	83.5	34.1	**76.6**	60.5	27.2	**84.2**	32.9	38.2	1.0	30.2	31.2	**43.6**

Table 4: The semantic segmentation performance on GTA5 ⇒ Cityscapes (19 common classes). We reported both per-category and mean IoU. All methods use ResNet101 as the backbone. '†': Results we obtained using the publicly released codes by the authors without any change (which is the baseline of our STAR).

Method	road	side.	buil.	light	sign	vege.	sky	pers.	rider	car	bus	motor	bike	**mIoU**
Source Only	55.6	23.8	74.6	6.1	12.1	74.8	79.0	55.3	19.1	39.6	23.3	13.7	25.0	38.6
ASN (feature) [54]	62.4	21.9	76.3	11.7	11.4	75.3	80.9	53.7	18.5	59.7	13.7	20.6	24.0	40.8
ASN (single-level) [54]	79.2	37.2	78.8	9.9	10.5	78.2	80.5	53.5	19.6	67.0	29.5	21.6	31.3	45.9
ASN (multi-level) [54]	**84.3**	**42.7**	77.5	4.7	7.0	77.9	**82.5**	54.3	21.0	72.3	32.2	18.9	32.3	46.7
CLAN [31]	81.3	37.0	80.1	**16.1**	**13.7**	78.2	81.5	53.4	21.2	73.0	32.9	**22.6**	30.7	47.8
CLAN†	77.0	30.8	**82.1**	8.6	10.8	**81.5**	80.2	56.9	21.6	71.6	29.2	11.6	36.3	46.0
STAR	82.6	36.2	81.1	12.2	8.7	78.4	82.2	**59.0**	**22.5**	**76.3**	**33.6**	11.9	**40.8**	**48.1**

Table 5: The semantic segmentation performance on Synthia ⇒ Cityscapes (13 common classes). We reported both per-category and mean IoU. All methods use ResNet101 as the backbone. '†': Results we obtained using the publicly released codes by the authors without any change (which is the baseline of our STAR).

4.2. Semantic Segmentation

Apart from image classification, we further evaluated our STAR on the semantic segmentation task which needs to classify every pixel of an image for understanding fine-grained details of the image content.

Datasets We used three popular semantic segmentation benchmarks in this experiment, namely GTA5 [40], Synthia [42] and Cityscapes [6]. Both GTA5 and Synthia are synthetic image datasets developed for avoiding the high cost of collecting dense pixel-level semantic annotations. GTA5 contains 24,966 images synthesised from an open-world computer game, whilst Synthia has 9,400 images generated as the random perturbation of virtual worlds. Cityscapes is a real street scene dataset (see Figure 5), including a training set of 2,975 images, a validation set of 500 images, and a testing set of 1,525 images. For a fair comparison, we used the validation set as the test set as in [31, 54]. We used one synthetic image dataset (GTA5 or Synthia) as the source domain data, and the real image dataset (Cityscapes) as the target domain.

Model instantialisation We used the ResNet101 based DeepLab-v2 [4] as the backbone. We selected the state-of-the-art CLAN [31] as the UDA framework. We removed CLAN's classifier weight discrepancy loss when constructing STAR since it has already been modelled by the variance of the STAR's distribution.

Implementation details For feature extractor, we used the SGD optimiser with a momentum of 0.9, the initial learning rate 2.5×10^{-4} in a polynomial decay with power of 0.9, and the weight decay 5×10^{-4}. For the classifier, we used the Adam [18] optimiser with β_1 = 0.9 and β_2 = 0.99, a fixed learning rate 5×10^{-5} and the weight decay 5×10^{-4}. We set the max training iterations to $100k$. The input images were cropped to $512 \times 1,024$ in training and up-sampled by a factor of 2 in test.

Results We evaluated the semantic segmentation performance of STAR in comparison with that of state-of-the-art methods in two UDA settings, GTA5 ⇒ Cityscapes (Table 4) and Synthia ⇒ Cityscapes (Table 5). All the compared methods use the same ResNet101 backbone. We have the following observations from the two tables: **(1)** As in image classification, the source trained model is inferior if directly applied to the target domain due to the domain shift problem. **(2)** The mIoU margins of STAR over the reported results of CLAN in [31] seem to be small (0.4% on GTA5⇒Cityscapes and 0.3% on Synthia⇒Cityscapes). However, when using the author-released code, we can never achieve the reported performance. A fairer comparison against our results using CLAN (CLAN†) with exactly the same hyperparameter setting shows a more substantial improvement (0.7% on GTA5⇒Cityscapes and 2.1% on Synthia⇒Cityscapes). **(3)** STAR achieves the best accuracy on both UDA settings, suggesting the overall performance advantage of the proposed method. To qualitatively examine the efficacy of our model, in Figure 5, we provided four

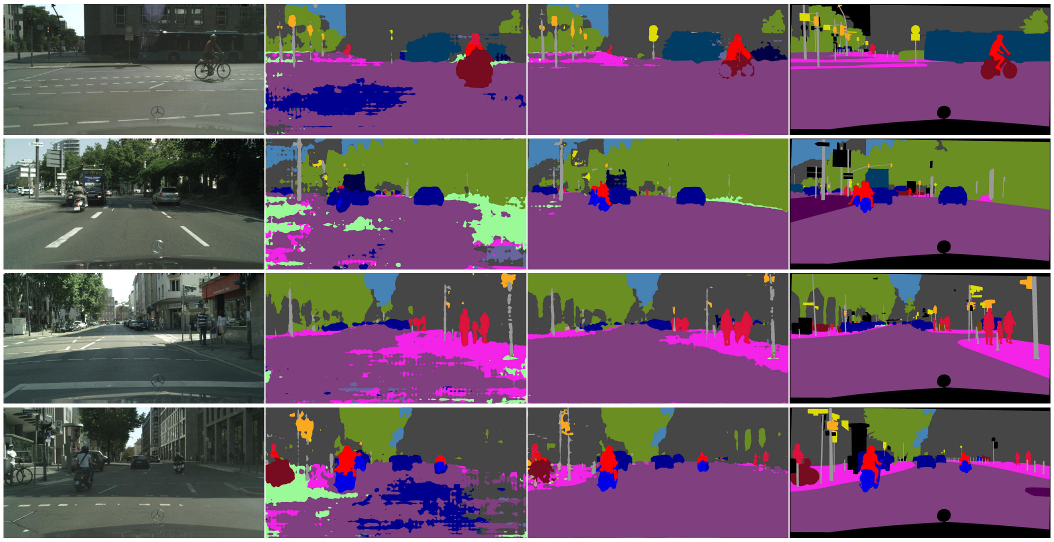


Figure 5: Qualitative semantic segmentation examples from the UDA setting of GTA5 [40] ⇒ Cityscapes [6]. Column **1**: Input images, Column **2**: Output of the source trained model, Column **3**: Output of our STAR, Column **4**: Ground truth.

randomly selected segmentation examples to visualise the performance boost of STAR in comparison to the source-trained model.

Variance function after convergence As discussed in Sec. 3.5, the classifier weight distribution (Gaussian) variances still have certain values when converging. Some of the variances are larger than others, deciding their specialised functions. The larger ones tend to render sampled classifiers diverse due to the wider sampling space, whilst the small ones guarantee the discrimination of classifiers on source domain by reducing effects on μ. With a joint effort from these two, STAR becomes more generalised on the target domain.

4.3. Ablation Study

The results reported so far suggest clearly that adding STAR to a local alignment based UDA method brings clear benefits. Here we examined the performance sensitivity of STAR against the sampled classifier number (two classifiers were sampled by default) at each training iteration of STAR. We did this test on two digit classification tasks (MNIST ⇒ USPS and USPS ⇒ MNIST), using MCD [45] as the UDA framework. For each specific classifier number, we repeated five times and reported the mean accuracy along with the standard deviation. Figure 6 shows that sampling more classifiers per iteration does not help to improve the performance, whilst introducing extra computational cost. This makes sense since at each iteration our STAR samples randomly *independent* classifiers, leading to using a large quantities of classifiers at the end of training. Hence, there is little incentive to sample more classifiers in a single iteration.

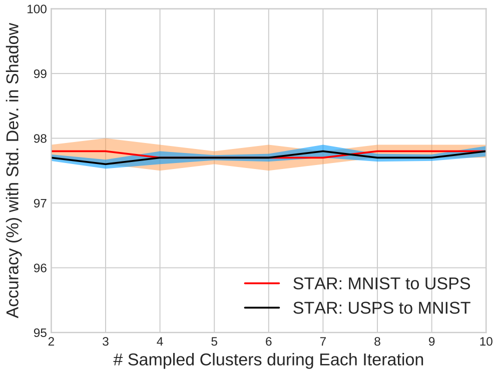


Figure 6: Performance sensitivity of our STAR against the classifier number per training iteration. We repeated each experiment five times and reported the average accuracy (solid line) with the standard deviation (shadow).

5. Conclusion

In this paper, we proposed STochastic clAssifieRs (STAR) for modelling diverse classifiers based on the observation that more classifiers perform better in UDA tasks. Compared with previous models utilising multiple classifiers for point-wise estimation, we build a multivariate Gaussian distribution $\mathcal{N}(\mu, \Sigma)$ over weights of classifiers. With it, naturally, an arbitrary number of diverse classifiers can be sampled. This enables us to take advantage of infinite classifiers without the increase in model size and the risks of overfitting. To show the general applicability of our method, we impose it onto two different pipelines for classification tasks and segmentation tasks respectively. From the results, STAR brings clear benefits and outperforms many state-of-the-art methods.

References

[1] Charles Blundell, Julien Cornebise, Koray Kavukcuoglu, and Daan Wierstra. Weight uncertainty in neural networks. In *ICML*, 2015. 3

[2] Konstantinos Bousmalis, Nathan Silberman, David Dohan, Dumitru Erhan, and Dilip Krishnan. Unsupervised pixel-level domain adaptation with generative adversarial networks. In *CVPR*, 2017. 3, 5

[3] Woong-Gi Chang, Tackgeun You, Seonguk Seo, Suha Kwak, and Bohyung Han. Domain-specific batch normalization for unsupervised domain adaptation. In *CVPR*, 2019. 3, 6

[4] Liang-Chieh Chen, George Papandreou, Iasonas Kokkinos, Kevin Murphy, and Alan L Yuille. Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. *TPAMI*, 2017. 7

[5] Xinyang Chen, Sinan Wang, Mingsheng Long, and Jianmin Wang. Transferability vs. discriminability: Batch spectral penalization for adversarial domain adaptation. In *ICML*, 2019. 6

[6] 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 *CVPR*, 2016. 7, 8

[7] Bharath Bhushan Damodaran, Benjamin Kellenberger, Rémi Flamary, Devis Tuia, and Nicolas Courty. Deepjdot: Deep joint distribution optimal transport for unsupervised domain adaptation. In *ECCV*, 2018. 5

[8] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *CVPR*, 2009. 6

[9] Carl Doersch. Tutorial on variational autoencoders. *arXiv preprint arXiv:1606.05908*, 2016. 3

[10] Yaroslav Ganin and Victor Lempitsky. Unsupervised domain adaptation by backpropagation. In *ICML*, 2015. 1, 4, 6

[11] Yaroslav Ganin, Evgeniya Ustinova, Hana Ajakan, Pascal Germain, Hugo Larochelle, François Laviolette, Mario Marchand, and Victor Lempitsky. Domain-adversarial training of neural networks. *Machine Learning Research*, 2016. 3, 5

[12] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *NeurIPS*, 2014. 3

[13] Philip Haeusser, Thomas Frerix, Alexander Mordvintsev, and Daniel Cremers. Associative domain adaptation. In *ICCV*, 2017. 5

[14] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, 2016. 1, 6

[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. 1, 2, 5

[16] Jonathan J. Hull. A database for handwritten text recognition research. *TPAMI*, 1994. 5

[17] Gareth James. Majority vote classifiers: theory and applications. 1998. 5

[18] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *ICLR*, 2015. 6, 7

[19] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. In *ICLR*, 2014. 3, 4

[20] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *NeurIPS*, 2012. 1

[21] Yann LeCun, Léon Bottou, Yoshua Bengio, Patrick Haffner, et al. Gradient-based learning applied to document recognition. *Proc.IEEE*, 1998. 5

[22] Chen-Yu Lee, Tanmay Batra, Mohammad Haris Baig, and Daniel Ulbricht. Sliced wasserstein discrepancy for unsupervised domain adaptation. In *CVPR*, 2019. 1, 2, 6

[23] Seungmin Lee, Dongwan Kim, Namil Kim, and Seong-Gyun Jeong. Drop to adapt: Learning discriminative features for unsupervised domain adaptation. In *ICCV*, 2019. 6

[24] Jingjing Li, Mengmeng Jing, Ke Lu, Lei Zhu, and Heng Tao Shen. Locality preserving joint transfer for domain adaptation. *TIP*, 2019. 6

[25] 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 *ECCV*, 2014. 6

[26] Ming-Yu Liu, Thomas Breuel, and Jan Kautz. Unsupervised image-to-image translation networks. In *NeurIPS*, 2017. 5

[27] Ming-Yu Liu and Oncel Tuzel. Coupled generative adversarial networks. In *NeurIPS*, 2016. 3, 5

[28] Mingsheng Long, Yue Cao, Jianmin Wang, and Michael I Jordan. Learning transferable features with deep adaptation networks. In *ICML*, 2015. 1, 2, 6

[29] Mingsheng Long, Han Zhu, Jianmin Wang, and Michael I Jordan. Unsupervised domain adaptation with residual transfer networks. In *NeurIPS*, 2016. 1, 2

[30] Mingsheng Long, Han Zhu, Jianmin Wang, and Michael I Jordan. Deep transfer learning with joint adaptation networks. In *ICML*, 2017. 6

[31] Yawei Luo, Liang Zheng, Tao Guan, Junqing Yu, and Yi Yang. Taking a closer look at domain shift: Category-level adversaries for semantics consistent domain adaptation. In *CVPR*, 2019. 1, 2, 3, 4, 7

[32] Boris Moiseev, Artem Konev, Alexander Chigorin, and Anton Konushin. Evaluation of traffic sign recognition methods trained on synthetically generated data. In *ICACIVS*, 2013. 5

[33] Radford M Neal. *Bayesian learning for neural networks*. Springer Science & Business Media, 2012. 3

[34] Yuval Netzer, Tao Wang, Adam Coates, Alessandro Bissacco, Bo Wu, and Andrew Y Ng. Reading digits in natural images with unsupervised feature learning. 2011. 5

[35] Yingwei Pan, Ting Yao, Yehao Li, Yu Wang, Chong-Wah Ngo, and Tao Mei. Transferrable prototypical networks for unsupervised domain adaptation. In *CVPR*, 2019. 6

[36] Vishal M Patel, Raghuraman Gopalan, Ruonan Li, and Rama Chellappa. Visual domain adaptation: A survey of recent advances. *Signal Processing*, 2015. 1

[37] Xingchao Peng, Ben Usman, Neela Kaushik, Judy Hoffman, Dequan Wang, and Kate Saenko. Visda: The visual domain adaptation challenge. *arXiv preprint arXiv:1710.06924*, 2017. 6

[38] Pedro O Pinheiro. Unsupervised domain adaptation with similarity learning. In *CVPR*, 2018. 5

[39] Can Qin, Lichen Wang, Yulun Zhang, and Yun Fu. Generatively inferential co-training for unsupervised domain adaptation. In *ICCV-W*, 2019. 5, 6, 7

[40] Stephan R Richter, Vibhav Vineet, Stefan Roth, and Vladlen Koltun. Playing for data: Ground truth from computer games. In *ECCV*, 2016. 7, 8

[41] Galina Rogova. Combining the results of several neural network classifiers. *Neural networks*, 1994. 2

[42] 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. 7

[43] Kuniaki Saito, Yoshitaka Ushiku, and Tatsuya Harada. Asymmetric tri-training for unsupervised domain adaptation. In *ICML*, 2017. 3, 4

[44] Kuniaki Saito, Yoshitaka Ushiku, Tatsuya Harada, and Kate Saenko. Adversarial dropout regularization. In *ICLR*, 2018. 3, 6

[45] Kuniaki Saito, Kohei Watanabe, Yoshitaka Ushiku, and Tatsuya Harada. Maximum classifier discrepancy for unsupervised domain adaptation. In *CVPR*, 2018. 1, 2, 3, 4, 5, 6, 8

[46] Swami Sankaranarayanan, Yogesh Balaji, Carlos D Castillo, and Rama Chellappa. Generate to adapt: Aligning domains using generative adversarial networks. In *CVPR*, 2018. 5

[47] Jian Shen, Yanru Qu, Weinan Zhang, and Yong Yu. Wasserstein distance guided representation learning for domain adaptation. In *AAAI*, 2018. 2

[48] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. In *ICLR*, 2015. 1

[49] Johannes Stallkamp, Marc Schlipsing, Jan Salmen, and Christian Igel. The german traffic sign recognition benchmark: A multi-class classification competition. In *IJCNN*, 2011. 5

[50] Mahesh Subedar, Ranganath Krishnan, Paulo Lopez Meyer, Omesh Tickoo, and Jonathan Huang. Uncertainty aware audiovisual activity recognition using deep bayesian variational inference. In *CVPR-W*, 2019. 3

[51] Quan-Sen Sun, Sheng-Gen Zeng, Yan Liu, Pheng-Ann Heng, and De-Shen Xia. A new method of feature fusion and its application in image recognition. *PR*, 2005. 5

[52] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *CVPR*, 2015. 1

[53] E Ke Tang, Ponnuthurai N Suganthan, and Xin Yao. An analysis of diversity measures. *Machine learning*, 2006. 3

[54] Yi-Hsuan Tsai, Wei-Chih Hung, Samuel Schulter, Kihyuk Sohn, Ming-Hsuan Yang, and Manmohan Chandraker. Learning to adapt structured output space for semantic segmentation. In *CVPR*, 2018. 3, 7

[55] Eric Tzeng, Judy Hoffman, Kate Saenko, and Trevor Darrell. Adversarial discriminative domain adaptation. In *CVPR*, 2017. 5

[56] Mei Wang and Weihong Deng. Deep visual domain adaptation: A survey. *Neurocomputing*, 2018. 1

[57] Ruijia Xu, Guanbin Li, Jihan Yang, and Liang Lin. Larger norm more transferable: An adaptive feature norm approach for unsupervised domain adaptation. In *ICCV*, 2019. 6

[58] Kaichao You, Ximei Wang, Mingsheng Long, and Michael Jordan. Towards accurate model selection in deep unsupervised domain adaptation. In *ICML*, 2019. 6

[59] Tianyuan Yu, Da Li, Yongxin Yang, Timothy M Hospedales, and Tao Xiang. Robust person re-identification by modelling feature uncertainty. In *ICCV*, 2019. 3

[60] Werner Zellinger, Thomas Grubinger, Edwin Lughofer, Thomas Natschläger, and Susanne Saminger-Platz. Central moment discrepancy (cmd) for domain-invariant representation learning. In *ICLR*, 2017. 2

[61] Zhi-Hua Zhou and Ming Li. Tri-training: Exploiting unlabeled data using three classifiers. *TKDE*, 2005. 3, 4