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].

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:
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 that is 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 computational 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.](image-url)
More specifically, we build a multivariate Gaussian distribution \( N(\mu, \Sigma) \), parametrised by a mean vector \( \mu \) and a diagonal covariance matrix \( \Sigma \). Whenever we need a number of classifiers, we can sample from \( N(\mu, \Sigma) \), and the relevant loss will be back-propagated to the learnable parameters \( \mu \) and \( \Sigma \). The choice of multivariate Gaussian for the classifier distribution is twofold: (i) it is reparametrisable [19], which is crucial for backpropagation; (ii) with diagonal \( \Sigma \), 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., \( \mu \) and \( \Sigma \), 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,

\[
\text{Step A } \min_{\theta, \phi_1, \phi_2} \ell_1(f_{\phi_1}(g_\theta(x_S)), y_S) + \ell_2(f_{\phi_2}(g_\theta(x_S)), y_S)
\]

\[
\text{Step B } \max_{\phi_1, \phi_2} \|f_{\phi_1}(g_\theta(x_T)) - f_{\phi_2}(g_\theta(x_T))\|_1
\]

\[
\text{Step C } \min_{\phi} \|f_{\phi_1}(g_\theta(x_T)) - f_{\phi_2}(g_\theta(x_T))\|_1
\]

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 \( N(\mu, \Sigma) \).

The sampling process is usually non-differentiable, thus we adapt the \textit{reparameterisation trick}, i.e., \( \hat{\phi}_1 = \mu + \sigma \odot \epsilon_1 \) and \( \hat{\phi}_2 = \mu + \sigma \odot \epsilon_2 \). Here \( \epsilon_1 \) and \( \epsilon_2 \) are two independent samples drawn from a standard Gaussian. \( \odot \) denotes element-wise product and \( \sigma \) is the diagonal of \( \Sigma \).

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.,

\[
\ell_{\theta, \phi_1, \phi_2, \psi}(x_S, x_T) = - \log(h_{\psi}(f_{\phi_1}(g_\theta(x_S)))) - \log(h_{\psi}(f_{\phi_2}(g_\theta(x_T)))) - \rho \log(1 - h_{\psi}(f_{\phi_1}(g_\theta(x_T)))) - \rho \log(1 - h_{\psi}(f_{\phi_2}(g_\theta(x_T))))
\]

and the min-max optimisation is built as,

\[
\min_{\psi} \max_{\theta, \phi_1, \phi_2} \ell_{\theta, \phi_1, \phi_2, \psi}(x_S, x_T)
\]

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., \( \rho = 1 - \frac{\phi_1^T \phi_2}{\|\phi_1\| \|\phi_2\|} \) 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.,

\[
\ell_{\psi}^{(W)} = \frac{\phi_1^T \phi_2}{\|\phi_1\| \|\phi_2\|}
\]

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

\textbf{Step 1} \( \min_{\theta, \phi_1, \phi_2} \ell_{\tilde{y}_{\psi}^{(1)}, y_S} + \ell_{\tilde{y}_{\psi}^{(2)}, y_S} + \ell_{(W)} - \ell_{(A)}(x_T) \)

\textbf{Step 2} \( \min_{\psi} \ell_{(A)}(x_S, x_T) \)

where \( \ell(\cdot, \cdot) \) is the segmentation loss (multi-class cross-entropy loss at pixel level), \( \tilde{y}_{S}^{(1)} = f_{\phi_1}(g_\theta(x_S)) \), \( \tilde{y}_{S}^{(2)} = f_{\phi_2}(g_\theta(x_S)) \) and \( \ell_{(A)}(x_T) \) 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 \( \epsilon_1 \) and \( \epsilon_2 \) 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
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 $\Sigma$. As we can see from Figure 4, the initial values of $\Sigma$ 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 classifiers 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.

![Figure 3: Toy experiments on the two-moon 2D dataset.](image)

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.

![Figure 4: The distribution of the flattened $\Sigma$ values for (a) initialisation (b) after convergence of STAR on MNIST to USPS task.](image)

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

<table>
<thead>
<tr>
<th>Method</th>
<th>SVHN ↓ MNIST</th>
<th>SYN SIG ↓ GTSRB</th>
<th>MNIST ↓ USPS</th>
<th>USPS ↓ MNIST</th>
</tr>
</thead>
<tbody>
<tr>
<td>Source only</td>
<td>67.1</td>
<td>85.1</td>
<td>79.4</td>
<td>63.4</td>
</tr>
<tr>
<td>DANN [11]</td>
<td>84.2</td>
<td>-</td>
<td>90.4</td>
<td>94.7</td>
</tr>
<tr>
<td>ADDA [55]</td>
<td>76.0±1.8</td>
<td>-</td>
<td>-</td>
<td>90.1±0.8</td>
</tr>
<tr>
<td>CoGAN [27]</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>89.1±0.8</td>
</tr>
<tr>
<td>PixDA [2]</td>
<td>-</td>
<td>-</td>
<td>95.9</td>
<td>-</td>
</tr>
<tr>
<td>ASSC [13]</td>
<td>95.7±1.3</td>
<td>82.8±1.3</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>UNIT [26]</td>
<td>90.5</td>
<td>-</td>
<td>96.0</td>
<td>93.6</td>
</tr>
<tr>
<td>Cy-CADA [15]</td>
<td>90.4±0.4</td>
<td>95.6±0.2</td>
<td>96.5±0.1</td>
<td>-</td>
</tr>
<tr>
<td>GTA [36]</td>
<td>92.4±0.9</td>
<td>95.3±0.7</td>
<td>90.8±1.3</td>
<td>-</td>
</tr>
<tr>
<td>DeepJDOT [7]</td>
<td>96.7</td>
<td>-</td>
<td>95.7</td>
<td>96.4</td>
</tr>
<tr>
<td>SimNet [38]</td>
<td>-</td>
<td>-</td>
<td>96.4</td>
<td>95.6</td>
</tr>
<tr>
<td>GiCt [39]</td>
<td>98.7</td>
<td>-</td>
<td>96.2</td>
<td>96.6</td>
</tr>
<tr>
<td>MCD [45]</td>
<td>96.2±0.4</td>
<td>94.4±0.3</td>
<td>96.5±0.3</td>
<td>94.1±0.3</td>
</tr>
<tr>
<td>STAR</td>
<td>98.8±0.05</td>
<td>95.8±0.2</td>
<td>97.8±0.1</td>
<td>97.7±0.05</td>
</tr>
</tbody>
</table>

![Figure 3: Toy experiments on the two-moon 2D dataset.](image)

**How STAR Works** The improved performance is credited to the variance of distribution $\Sigma$. As we can see from Figure 4, the initial values of $\Sigma$ 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 classifiers 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.

**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 $\mu$ 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 handwriting..
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.

| Method     | plane | bcycl | bus | car | horse | knife | mcycl | person | plant | sktbrd |Work on this task can be found in [14].

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 identify 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 instantiation 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 $2 \times 10^{-4}$ and the batch size 128. We trained 200 epochs for SVHN $\Rightarrow$ MNIST and SYNSIGN $\Rightarrow$ GTSRB, and 300 epochs for USPS $\Rightarrow$ MNIST and MNIST $\Rightarrow$ 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 identify 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 instantiation For a fair comparison, 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 $224 \times 224$. We adopted the SGD optimiser with the batch size 32, the learning rate $1.0 \times 10^{-3}$ 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\textsuperscript{1} We did not experiment with SWD [22] since we cannot reproduce the reported results even with the help of the authors.
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 instantisation 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 $2.5 \times 10^{-4}$ in a polynomial decay with power of 0.9, and the weight decay $5 \times 10^{-4}$. For the classifier, we used the Adam [18] optimiser with $\beta_1 = 0.9$ and $\beta_2 = 0.99$, a fixed learning rate $5 \times 10^{-5}$ and the weight decay $5 \times 10^{-4}$. We set the max training iterations to $100k$. The input images were cropped to $512 \times 1024$ 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

<table>
<thead>
<tr>
<th>Method</th>
<th>road</th>
<th>side</th>
<th>bul.</th>
<th>wall</th>
<th>fence</th>
<th>pole</th>
<th>light</th>
<th>sign</th>
<th>vgg</th>
<th>veg</th>
<th>terr.</th>
<th>sky</th>
<th>car</th>
<th>truck</th>
<th>bike</th>
<th>mIoU</th>
</tr>
</thead>
<tbody>
<tr>
<td>Source Only</td>
<td>75.8</td>
<td>16.8</td>
<td>77.2</td>
<td>12.5</td>
<td>21.0</td>
<td>25.5</td>
<td>30.1</td>
<td>20.1</td>
<td>81.3</td>
<td>24.6</td>
<td>70.3</td>
<td>53.8</td>
<td>26.4</td>
<td>49.9</td>
<td>17.2</td>
<td>25.9</td>
</tr>
<tr>
<td>ASN (feature) [54]</td>
<td>83.7</td>
<td>27.6</td>
<td>75.5</td>
<td>20.3</td>
<td>19.9</td>
<td>27.4</td>
<td>28.3</td>
<td>27.4</td>
<td>79.0</td>
<td>28.4</td>
<td>70.1</td>
<td>55.1</td>
<td>20.2</td>
<td>72.9</td>
<td>22.5</td>
<td>35.7</td>
</tr>
<tr>
<td>ASN (single-level) [54]</td>
<td>86.5</td>
<td>25.9</td>
<td>79.8</td>
<td>22.1</td>
<td>20.0</td>
<td>23.6</td>
<td>33.1</td>
<td>21.8</td>
<td>81.8</td>
<td>25.9</td>
<td>75.9</td>
<td>57.3</td>
<td>26.2</td>
<td>76.3</td>
<td>29.8</td>
<td>32.1</td>
</tr>
<tr>
<td>ASN (multi-level) [54]</td>
<td>86.5</td>
<td>36.0</td>
<td>79.9</td>
<td>23.4</td>
<td>23.3</td>
<td>23.9</td>
<td>35.2</td>
<td>14.8</td>
<td>83.4</td>
<td>33.3</td>
<td>75.6</td>
<td>58.5</td>
<td>27.6</td>
<td>73.7</td>
<td>32.5</td>
<td>35.4</td>
</tr>
<tr>
<td>GICT [39]</td>
<td>88.6</td>
<td>41.3</td>
<td>76.4</td>
<td>23.3</td>
<td>26.1</td>
<td>24.3</td>
<td>32.8</td>
<td>23.1</td>
<td>82.3</td>
<td>37.4</td>
<td>73.3</td>
<td>62.2</td>
<td>24.8</td>
<td>73.3</td>
<td>39.6</td>
<td>4.6</td>
</tr>
<tr>
<td>CLAN [31]</td>
<td>87.0</td>
<td>27.1</td>
<td>79.6</td>
<td>27.3</td>
<td>23.3</td>
<td>28.3</td>
<td>35.5</td>
<td>24.2</td>
<td>83.6</td>
<td>27.4</td>
<td>74.2</td>
<td>58.6</td>
<td>28.0</td>
<td>76.2</td>
<td>33.1</td>
<td>31.4</td>
</tr>
<tr>
<td>CLAN†</td>
<td>87.9</td>
<td>28.6</td>
<td>79.9</td>
<td>24.0</td>
<td>24.0</td>
<td>24.9</td>
<td>33.3</td>
<td>19.4</td>
<td>83.4</td>
<td>31.9</td>
<td>76.1</td>
<td>58.3</td>
<td>27.5</td>
<td>82.8</td>
<td>35.4</td>
<td>42.6</td>
</tr>
<tr>
<td>STAR</td>
<td>88.4</td>
<td>27.9</td>
<td>80.8</td>
<td>27.3</td>
<td>25.6</td>
<td>26.9</td>
<td>31.6</td>
<td>20.8</td>
<td>83.5</td>
<td>34.1</td>
<td>76.6</td>
<td>60.5</td>
<td>27.2</td>
<td>84.2</td>
<td>32.9</td>
<td>38.2</td>
</tr>
</tbody>
</table>

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).

<table>
<thead>
<tr>
<th>Method</th>
<th>road</th>
<th>side</th>
<th>bul.</th>
<th>wall</th>
<th>fence</th>
<th>pole</th>
<th>light</th>
<th>sign</th>
<th>vgg</th>
<th>veg</th>
<th>terr.</th>
<th>sky</th>
<th>car</th>
<th>truck</th>
<th>bike</th>
<th>mIoU</th>
</tr>
</thead>
<tbody>
<tr>
<td>Source Only</td>
<td>55.6</td>
<td>23.8</td>
<td>74.6</td>
<td>6.1</td>
<td>12.1</td>
<td>74.8</td>
<td>79.0</td>
<td>55.3</td>
<td>19.1</td>
<td>39.6</td>
<td>23.3</td>
<td>13.7</td>
<td>25.0</td>
<td>38.6</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ASN (feature) [54]</td>
<td>62.4</td>
<td>21.9</td>
<td>76.3</td>
<td>11.7</td>
<td>11.4</td>
<td>75.5</td>
<td>80.9</td>
<td>53.7</td>
<td>18.5</td>
<td>59.7</td>
<td>13.7</td>
<td>20.6</td>
<td>24.0</td>
<td>40.8</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ASN (single-level) [54]</td>
<td>79.2</td>
<td>37.2</td>
<td>78.8</td>
<td>9.9</td>
<td>10.5</td>
<td>78.2</td>
<td>80.5</td>
<td>53.5</td>
<td>19.6</td>
<td>67.0</td>
<td>29.5</td>
<td>21.6</td>
<td>31.3</td>
<td>45.9</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ASN (multi-level) [54]</td>
<td>84.3</td>
<td>42.7</td>
<td>77.5</td>
<td>4.7</td>
<td>7.0</td>
<td>77.9</td>
<td>82.5</td>
<td>54.3</td>
<td>21.0</td>
<td>72.3</td>
<td>32.2</td>
<td>18.9</td>
<td>32.3</td>
<td>46.7</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CLAN [31]</td>
<td>81.3</td>
<td>37.0</td>
<td>80.1</td>
<td>16.1</td>
<td>13.7</td>
<td>78.2</td>
<td>81.5</td>
<td>53.4</td>
<td>21.2</td>
<td>73.0</td>
<td>32.9</td>
<td>22.6</td>
<td>30.7</td>
<td>47.8</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CLAN†</td>
<td>77.0</td>
<td>30.8</td>
<td>82.1</td>
<td>8.6</td>
<td>10.8</td>
<td>81.5</td>
<td>80.2</td>
<td>56.9</td>
<td>21.6</td>
<td>71.6</td>
<td>29.2</td>
<td>11.6</td>
<td>36.3</td>
<td>46.0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>STAR</td>
<td>82.6</td>
<td>36.2</td>
<td>81.1</td>
<td>12.2</td>
<td>8.7</td>
<td>78.4</td>
<td>82.2</td>
<td>59.0</td>
<td>22.5</td>
<td>76.3</td>
<td>33.6</td>
<td>11.9</td>
<td>40.8</td>
<td>48.1</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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).
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 $\mu$. 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 $\Rightarrow$ USPS and USPS $\Rightarrow$ 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.

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


[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


[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


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


