Certainty Volume Prediction for Unsupervised Domain Adaptation

Tobias Ringwald
tobias.ringwald@kit.edu
Rainer Stiefelhagen
rainer.stiefelhagen@kit.edu
Institute for Anthropomatics and Robotics (CV:HCI Lab)
Karlsruhe Institute of Technology
Karlsruhe, Germany

Abstract

Unsupervised domain adaptation (UDA) deals with the problem of classifying unlabeled target domain data while labeled data is only available for a different source domain. Unfortunately, commonly used classification methods cannot fulfill this task adequately due to the domain gap between the source and target data. In this paper, we propose a novel uncertainty-aware domain adaptation setup that models uncertainty as a multivariate Gaussian distribution in feature space. We show that our proposed uncertainty measure correlates with other common uncertainty quantifications and relates to smoothing the classifier’s decision boundary, therefore improving the generalization capabilities. We evaluate our proposed pipeline on challenging UDA datasets and achieve state-of-the-art results. Code for our method is available at https://gitlab.com/tringwald/cvp.

1 Introduction

In recent years, deep neural classification networks have been successfully applied to a wide variety of tasks. One key contribution to that success is the utilization of enormous amounts of labeled training data during the optimization process of such networks. While abundant labeled data might be readily available for a given source domain, more specialized data is often hard to obtain or too expensive to manually label for a desired target domain (e.g. medical data). Thus, past research [19, 21, 28, 34, 36, 38] proposed to generate synthetic data for the training of classifiers, as this would render the manual labeling process unnecessary. Unfortunately, current classification architectures cannot handle the domain shift between the source and target domain, leading to a significantly degraded classification accuracy.

Unsupervised domain adaptation (UDA) seeks to address this domain shift problem by adapting a model to the desired target domain while only given labeled data of the source domain. Recent work has tackled this problem from multiple different angles: Holistic image-level and pixel-level methods often try to manipulate the global appearance of a given source or target domain image, for example by applying style transfer or other image-to-image mapping algorithms [1, 19]. At the feature-level, prior methods commonly focus on minimizing domain divergence measures [21, 28] or enforce source and target features to be indistinguishable through the means of adversarial training [3, 28].

© 2021. The copyright of this document resides with its authors.
It may be distributed unchanged freely in print or electronic forms.
However, these methods often neglect the intrinsic uncertainty accompanying the unsupervised domain adaptation process: as no target domain labels are available, pseudo-labels have to be constructed, which is an inherently noisy task involving uncertainty. In this paper, we thus propose a novel feature-level unsupervised domain adaptation algorithm called Certainty Volume Prediction (CVP). CVP integrates uncertainty into the domain adaptation pipeline by modelling a multivariate Gaussian distribution around a given feature. The covariance matrix of said distribution is constructed in a way that reflects the model’s current certainty (or uncertainty) for a feature given the current training step. We achieve this by enforcing samples from the distribution to belong to the same class as the original feature. Therefore, features for which the model is more certain can be allotted higher variance values than uncertain features, due to their distance from the decision boundary.

Finally, we show that our proposed feature space method correlates with other common uncertainty measures in classifier space – such as Monte-Carlo Dropout [8] – and relates to smoothing the decision boundary [2, 25, 39] which has been linked to better generalization and robustness properties [18]. In summary, our contributions are threefold: (i) We introduce our novel Certainty Volume Prediction algorithm for measuring uncertainty in feature space and show how it can help to improve a model’s domain adaptation capabilities. (ii) We show its correlation to other uncertainty measures such as Monte-Carlo Dropout. (iii) We show CVP’s state-of-the-art performance on multiple challenging datasets and link its increased generalization performance to the smoothness of the decision boundary.

2 Related Work

Recent works in the area of domain adaptation have viewed the domain shift problem from multiple different angles: At a holistic image-level (often referred to as pixel-level), methods often try to find a common input or latent space in order to reuse the same classifier for both source and target domain. This can be achieved by the means of style transfer [1] and related image-to-image translation methods [19, 29]. A domain-agnostic space can also be created at the feature-level instead of the image-level. For this, Ganin et al. propose their gradient
reversal layer (RevGrad) in conjunction with a domain classifier, which enforces features from the source and target domain to be indistinguishable. Additionally, distribution divergence measures have been applied at the feature-level for the alignment of the source and target domain distributions: Long et al. \cite{26} employ a maximum mean discrepancy (MMD) \cite{13} based method in order to jointly adapt the marginal and conditional distributions. This idea of MMD-based optimization has recently been picked up again by Kang et al. \cite{21} for their proposed CDD loss and is also used for regularization in the PUnDA framework by Gholami et al. \cite{10}. Motivated by the results of Szegedy and Goodfellow \cite{12, 43}, adversarial approaches have also been investigated. In this subarea, Park et al. \cite{31} have applied adversarial dropout to improve generalization while Pinheiro \cite{34} proposes a categorical prototype-based domain-adversarial learning setup. Another orthogonal research directions was pursued by Chang et al. \cite{3}, who aim at capturing the different source and target distributions in domain-specific batch-normalization layers.

In this work, we focus on the integration of uncertainty into the domain adaptation pipeline. Related to this topic, Han et al. \cite{16} proposed to calibrate the predictive uncertainty on the target domain data through the source domain uncertainties. They model this uncertainty in the classifier space and quantify it as the predicted Rényi entropy on the target domain data. Guan et al. \cite{15} propose the UaDAN framework that employs conditional adversarial learning for alignment of samples. They measure uncertainty as a proxy metric based on the alignment quality of samples and use this in the alignment process as well as their proposed curriculum learning setup. Furthermore, Ringwald et al. \cite{37} propose the UFAL architecture, which measures uncertainty through Monte-Carlo Dropout (MCD) in classifier space. They then utilize this uncertainty for alignment of features in an Euclidean feature space and also for a curriculum learning-like filtering step.

Unlike the aforementioned approaches, our Certainty Volume Prediction (CVP) framework does not use post hoc measures for uncertainty but instead directly models uncertainty as a multivariate normal distribution in feature space, which is integrated in the backpropagation path of our model and can therefore be optimized during training. We show that our proposed uncertainty measure in feature space correlates with commonly used uncertainty measures in classifier space (such as Monte-Carlo Dropout) and can also be interpreted with regard to intuitive key assumptions. Finally, we show that CVP’s improved generalization performance can be linked to the smoothness of the classifier’s decision boundary.

3 Methodology

3.1 Basic Setup and Naming Conventions

In an unsupervised domain adaptation (UDA) setup, labeled instances are only available for the source domain while the target domain purely consists of unlabeled data. Formally, we consider the source domain dataset \(D_S = \{(x_S^{(i)}, y_S^{(i)})\}_{i=1}^{N_S}\) to contain \(N_S\) pairs of images \(x_S^{(i)}\) and labels \(y_S^{(i)} \in C\) where \(C = \{1, 2, \ldots, N_C\}\) is the set of classes with size \(|C| = N_C\), while the target domain dataset \(D_T = \{(x_T^{(i)})\}_{i=1}^{N_T}\) only consists of \(N_T\) unlabeled instances. The objective of an UDA algorithm is then to produce accurate predictions \(y_T^{(i)}\) for every target domain instance \(x_T^{(i)}\).

The UDA approach presented in this paper is based on a common convolutional neural network (CNN) feature extractor, which we denote as \(f_{\text{CNN}}\). The extracted features are
subsequently processed by our proposed feature sampling stage and a classifier $f_{C1}$, which maps its input into a probability distribution over the class set $C$. A basic overview of the full architecture is provided in Figure 1. Given these prerequisites, we will introduce our full CVP architecture in the following sections.

### 3.2 Advantages of Feature Space Uncertainty Estimation

The goal of our proposed Certainty Volume Prediction method is to model uncertainty in feature space in a way that improves the generalization process necessary for domain adaptation. This is in contrast to prior approaches [8, 16, 37] that model and measure uncertainty in the classifier-space. For example, Monte Carlo dropout [3] proposes to conduct $T$ forward passes at testing time using different dropout [41] masks. This results in $T$ different probability distributions over the class set $C$ which can then be used to calculate a model’s predictive mean and variance. Both of these values can be interpreted as a form of uncertainty measure. However, we argue that uncertainty should be estimated earlier in a model’s pipeline, as the classifier output has already been reduced to a much smaller, less informative space through mapping into class-space over $C$ and squashing by softmax normalization. Thus, we propose to use the feature space for the uncertainty estimation.

### 3.3 Modeling Uncertainty in Feature Space with CVP

CNN feature extractors provide a means to map any input image to a compact feature representation. One key property of this mapping process is that similar inputs should be mapped to similar outputs, i.e. images of the same class are mapped to features that have a small distance in feature space. This property is commonly exploited by unsupervised algorithms such as clustering and can also be observed in high-dimensional spaces through dimension reduction techniques (cf. [27]). Hence, it is often referred to as cluster assumption [4] and states that decision boundaries should not cross high density regions.

Our CVP algorithm also utilizes this property: We first apply a feature extractor $f_{CNN}$ to a given input image $x^{(i)}$ in order to compute a compact feature embedding $\mu^{(i)}$. Given above cluster assumption, features near $\mu^{(i)}$ should belong to the same class as $\mu^{(i)}$ itself. However, this leads to the question how to define what “near” means and how to obtain these features. In this paper, we propose to phrase proximity as an uncertainty estimation in feature space based on the decision boundary and visualize this in Figure 2. To achieve this, we model a multivariate Gaussian distribution around the predicted mean $\mu^{(i)}$. We employ a subnetwork $f_\Sigma$ (see Figure 1) to predict a suitable covariance matrix $\Sigma^{(i)}$, which enables us to sample related features from the resulting distribution $\mathcal{N}(\mu^{(i)}, \Sigma^{(i)})$. Given a $D$-dimensional $\mu$ feature, the matching covariance matrix would be of shape $D \times D$. However, for common CNN feature extractors, $D$ is often very large (e.g. $D = 2048$ for ResNet-50 [17]) which would result in a covariance matrix with millions of values and an even larger amount of weights in $f_\Sigma$ to predict those values. We thus propose to use a computationally feasible subnetwork $f_\Sigma: \mu^{(i)} \in \mathbb{R}^D \rightarrow \sigma^{(i)} \in \mathbb{R}$ and then sample from the distribution $\mathcal{N}(\mu^{(i)}, (\sigma^{(i)}\mathbb{1})^2)$ where $\mathbb{1}$ denotes the identity matrix.

Intuitively, features far from a decision boundary are located in an area of high certainty, i.e. many of the surrounding feature belong to the same class. We visualize this in Figure 2a for a 2-dimensional case: the distribution centered on $\mu^{(1)}$ can be assigned a large $\sigma$ without crossing a decision boundary, as it is located in such an area of high certainty. Features very close to the decision boundary must not have a large $\sigma$ (see $\mu^{(2)}$) as this might cross the
decision boundary and violates above assumption that nearby features should belong to the same classes (see $\mu^{(3)}$). Intuitively, it follows that $\sigma$ can be interpreted as an uncertainty estimate for a given input, where small $\sigma$ values signal uncertainty while large $\sigma$ values indicate certainty. The objective of our Certainty Volume Prediction setup is then to assess what area – or volume in higher dimensions – can be safely spanned by the distribution so that the majority of samples will not cross a decision boundary.

### 3.4 Optimization of CVP

We meet CVP’s objective defined above with the help of two loss functions. After obtaining $\mu^{(i)}$ and $\sigma^{(i)}$ given the above description, we draw $M$ feature samples from the resulting distribution $\varphi_{1:M}^{(i)} \sim \mathcal{N}(\mu^{(i)}, (\sigma^{(i)})^2)$. In order to satisfy the cluster assumption, we enforce all of these new features to belong to the same class $y^{(i)}$ as $\mu^{(i)}$. For this, we use the cross-entropy-based samples loss $L_{ce}^{\varphi}$ (see Equation 1) where $\hat{y}^{(i)}$ is the one-hot encoded label corresponding to $y^{(i)}$. Applying this loss will contract $\sigma^{(i)}$ if the certainty estimation is too high, i.e. samples are drawn too far from the mean and thus misclassified which increases the cross-entropy term. However, using the $L_{ce}^{\varphi}$ loss alone would collapse $\sigma^{(i)}$ towards 0 as this would minimize the objective function. Therefore, we also propose an additional antagonistic loss $L_{\text{Ant}}$ (see Equation 3). $L_{\text{Ant}}$ uses the $L_{ce}^{\varphi}$-based variable $\psi^{(i)}$ (see Equation 2) in order to enforce an inversely proportional relationship between the samples loss $L_{ce}^{\varphi}$ and the (un-)certainty estimation $\sigma^{(i)}$: The higher $L_{ce}^{\varphi}$ becomes, the lower $\sigma^{(i)}$ should be; at the maximum expected loss value for $L_{ce}^{\varphi}$ (denoted as $\kappa$), the target value $\psi^{(i)}$ for $\sigma^{(i)}$ should be 0. We finally model this as a regression task utilizing a smooth $\ell_1$ loss between $\sigma^{(i)}$ and $\psi^{(i)}$ (see Equation 3). We provide a theorem on how to derive the constant $\kappa$ in Appendix B.

$$L_{ce}^{\varphi}(\hat{y}^{(i)}, \varphi^{(i)}) = -\frac{1}{M} \sum_{m=1}^{M} \sum_{j=1}^{|C|} \hat{y}_j^{(i)} \log(f_{Cl}(\varphi_m^{(i)}))_j$$

$$\psi^{(i)}(\hat{y}^{(i)}, \varphi^{(i)}) = \max(\{0, \kappa - L_{ce}^{\varphi}(\hat{y}^{(i)}, \varphi^{(i)})\})$$

$$L_{\text{Ant}}(\sigma^{(i)}, \hat{y}^{(i)}, \varphi^{(i)}) = \begin{cases} \frac{1}{2} (\sigma^{(i)} - \psi^{(i)})^2 & \text{if } |\sigma^{(i)} - \psi^{(i)}| < 1 \\ |\sigma^{(i)} - \psi^{(i)}| & \text{otherwise} \end{cases}$$

Finally, we add the last loss term $L_{ce}^{\mu} = -\sum_{j=1}^{|C|} \hat{y}_j^{(i)} \log(f_{Cl}(\mu^{(i)}))_j$, which is equivalent to the normal feature classification in a standard CNN training setup. We thus arrive at the total loss $L_{\text{total}} = L_{ce}^{\mu} + \alpha L_{ce}^{\varphi} + L_{\text{Ant}}$ where $\alpha$ is a tuneable loss weight.

### 3.5 Training Process

Given above formal introduction of our proposed CVP algorithm, we can now describe our full domain adaptation pipeline. Overall, our setup is two-stage with an initial pretraining phase and an iterative domain adaptation refinement step.

**Pretraining Phase.** Our training setup starts with a pretraining phase. During this phase, only labeled source domain data is used. This phase mainly serves the purpose of initializing
Figure 2: From left to right: (a) Visualization of CVP’s interaction with the decision boundary. (b) A large amount of correctly classified instances will result in a small $L_c^{\phi}$ value and therefore an increase of $\sigma$. (c) Likewise, a large amount of incorrect classifications will increase $L_c^{\phi}$ and lower the training target $\psi$ and hence also $\sigma$.

the weights of the $f_\Sigma$ and $f_{CI}$, which can be reliably done with the ground truth source domain annotations. At the end of the pretraining phase, the combination of $f_\Sigma(f_{CNN}(\cdot))$ can already be used to estimate uncertainty while $f_{CI}(f_{CNN}(\cdot))$ can be employed as a pseudo-labeler. 

Adaptation Phase. The adaptation phase is an iterative cycle-based refinement step, during which the model is adapted to the target domain. At the start of every adaptation cycle, we utilize the current training progress of our model in order to create pseudo-labels for target domain instances, i.e. we use $\text{argmax} \circ f_{CI} \circ f_{CNN} : x_T^{(i)} \rightarrow y_T^{(i)}$ to generate a label $y_T^{(i)} \in C$ for every target domain instance $x_T^{(i)}$. We then follow the procedure of [37] and sample mixed domain batches consisting of 50% source domain and 50% target domain images for training. Given such a batch, the model is then optimized w.r.t. the loss function $L_{\text{total}}$ introduced in section 3.4. This is repeated for $T_{\text{cycle}}$ steps, until the next adaptation cycle begins. The pseudo-label generation always uses the current model weights, therefore only one set of model weights needs to be kept, i.e. no additional memory or storage space is necessary.

4 Experiments

4.1 Evaluation Setup

Datasets. We evaluate our proposed CVP method on four publicly available UDA benchmark datasets: VisDA-2017 (also known as Syn2Real-C) [33] is a staple dataset for unsupervised domain adaptation that offers over 200,000 images from 12 object classes in the domains synthetic and real. The synthetic domain (train set) contains 152,397 images of 3D renderings under different lighting conditions and viewing angles. The real domain (test set) contains 72,372 real life images extracted from the YouTube Bounding-Boxes dataset [35]. Adaptiope [38] is a very challenging UDA dataset with 123 classes in the three domains synthetic, product and real life. Adaptiope was constructed as a balanced dataset. Therefore, every class and every domain contains the same amount of images, resulting in 36,900 images overall with 100 images per class and 12,300 images per domain. Office-Caltech [38] is a common UDA dataset containing 2,533 images of 10 classes in the 4 domains Amazon, Caltech, DSLR and Webcam. Modern Office-31 [38] was recently proposed as a replacement
for the Office-31 [40] dataset. It rectifies the annotation errors in Office-31 and also adds a new synthetic domain for a total of 6,712 images of 31 classes in the 3 domains Amazon, Webcam and Synthetic. Example images from the datasets are shown in Figure 3.

Evaluation Metrics. For the evaluation of our method, we employ the standard metrics of each dataset: For Adaptiope, we construct six transfer tasks based on the three domains and calculate a mean and standard deviation over multiple runs as proposed in [38]. For VisDA-2017, we calculate the mean class accuracy as per challenge evaluation protocol. Here, the standard accuracy is computed per class and then averaged over all 12 classes for a final score. For Modern Office-31 and Office-Caltech, we construct 6 and 12 transfer tasks respectively and report the accuracy per task as well as an overall average. For the evaluation of the decision boundary smoothness, we employ the oscillation of classification metric from [22]. Here, we generate subfeatures by linearly interpolating between two features $\mu^{(i)}$ and $\mu^{(j)}$ (denoted as $\mu^{(i,j)}_k$ with $k$ being the interpolation step) and measure the number of changes in classification. Formally, we measure $O(\mu^{(i)}, \mu^{(j)}) = \frac{1}{K} \sum_{k=1}^{K-1} 1(\argmax f_{\mathcal{C}}(\mu^{(i,j)}_k) \neq \argmax f_{\mathcal{C}}(\mu^{(i,j)}_{k+1}))$ where 1 denotes the indicator function and $K$ is the number of interpolation steps. This metric is visualized in Figure 3c for $K = 6$. We provide further information about the implementation details and the evaluation setup in Appendix A.

4.2 Results

Analysis. We start evaluating our CVP method by providing an analysis w.r.t. CVP’s $\sigma$ prediction. In Figure 4a and b, we show the batch median $\sigma$ for both source and target domain samples during the adaptation phase as training progresses. During the pretraining phase, the model has only been trained on source domain data. For this reason, a large drop of $\sigma$ can be observed at the beginning of the adaptation phase, as the model is now confronted with target domain data for the first time. The resulting decrease in $\sigma$ can be interpreted as a lower certainty in the model’s predictions due to the involvement of new and unseen data. While the certainty on the source domain quickly recovers, the target domain certainty

Figure 3: (a) Example images from Adaptiope [38] with 123 classes in the domains product, real and synthetic. (b) Example images from VisDA-2017 [33] with 12 classes in the domains synthetic and real. (c) Visualization of the boundary smoothness evaluation. Top: Interpolating between $\mu^{(i)}$ and $\mu^{(j)}$ results in few class changes due to the smooth decision boundary. Bottom: Interpolating between $\mu^{(m)}$ and $\mu^{(n)}$ results in many changes due to the curvy shape of the boundary.
increase is delayed due to the domain gap and the ongoing domain adaptation during training. For the S→P task, our CVP model converges to a similar certainty (i.e. magnitude of $\sigma$) for both domains. However, for the hardest transfer task of Adaptope – S→R – the mean target domain certainty at the end of training is still lower than the source domain certainty due to the enormous domain gap of synthetic to real transfer tasks.

Furthermore, we analyze $\sigma$ w.r.t. common uncertainty measures. In total, we compare with five common uncertainty measures: (1) magnitude of the maximum logit ($L$), (2) magnitude of the logit for the ground truth class ($L_{GT}$), (3) the difference of the top two logits ($L_{diff}$), (4) the probability at the maximum logit prediction after averaging over multiple Monte Carlo Dropout (MCD) iterations ($MCD_\mu$) and (5) the standard deviation at the maximum logit prediction after multiple MCD iterations ($MCD_\sigma$, inverted for visualization purposes). We then calculate the Pearson correlation coefficient between $\sigma$ and measure 1-5 and plot the results in Figure 4c for multiple transfer tasks. For all measures we compare with, we observe a strong linear correlation indicating that $\sigma$ can indeed be interpreted as an (un-)certainty measure. This is also consistent for all evaluated transfer tasks.

We also conduct an ablation study w.r.t. our model’s three loss functions and show results in Table 1 for the four difficult transfer tasks involving Adaptope’s synthetic domain. In the basic setup, only $L_{ce}$ is used. This is the minimal setup in order to train the classifier and obtain pseudo-labels. As expected, this performs the worst of all evaluated setups, as CVP’s additional loss functions are not active. We continue by adding $L_{ant}$. This already improves the classification accuracy for all four transfer tasks. However, without the $L_{phi}$ loss, the cluster assumption is not enforced, which yields excessive $\sigma$ values and results in lower accuracies than our full CVP method. Only the usage of all three losses – $L_{ce}$, $L_{phi}$ and $L_{ant}$ – results in the highest accuracy for all four transfer tasks with an average improvement of almost 3% over the basic setup. Note that training with $L_{phi}$ but without $L_{ant}$ would instantly

| Metric Setup   | Losses | P→S | R→S | S→P | S→R | Avg. |
|----------------|--------|-----|-----|-----|-----|------|
| **Accuracy**   | Basic  | $L_{ce}$ | 69.4 | 70.1 | 74.8 | 60.0 | 68.6 |
|                | No Samples CE | $L_{ce}$ | 70.7 | 70.8 | 77.1 | 61.9 | 70.1 |
|                | Full CVP Setup | $L_{ce}$, $L_{ant}$ | 71.9 | 72.4 | 78.4 | 63.2 | 71.5 |
| **Oscillation**| Basic  | $L_{ce}$ | 205.9±1.3 | 207.2±0.2 | 221.2±3.8 | 259.6±2.7 | 223.5 |
|                | Full CVP Setup | $L_{ce}$, $L_{ant}$, $L_{phi}$ | 197.1±0.6 | 199.3±1.2 | 208.9±1.6 | 233.0±4.8 | 209.6 |

Table 1: Effect of CVP on the classification results and the smoothness of the decision boundary quantified by the oscillation of classification metric. Results are given as accuracy (in %, ↑ higher is better) and oscillation of classification metric $O(\cdot, \cdot)$ summed over all sampled feature pairs (↓ lower is better).
capabilities by smoothing the decision boundary of a classifier. For further ablation studies, we can see that CVP can be seen as an intuitive uncertainty measure that improves domain adaptation significantly. For instance, on the hardest transfer task S → R, we observe a 3.2% increase on the mean accuracy. On average, CVP improves the domain adaptation results by 2.9%, with an overall improvement across all transfer tasks. We also provide domain adaptation results with and without CVP in Table 1. We observe that every model employing CVP improves in accuracy on the target domain when compared to a model without CVP. On average, CVP improves the domain adaptation results by 2.9%, with a 3.2% increase on the hardest transfer task S → R and up to 3.6% on S → P. We thus conclude that CVP can be seen as an intuitive uncertainty measure that improves domain adaptation capabilities by smoothing the decision boundary of a classifier. For further ablation studies, please refer to Appendix C.

Comparison to State-of-the-Art. We now compare our proposed CVP algorithm to recently proposed unsupervised domain adaptation methods. On the VisDA-2017 dataset (see Table 2), CVP is able to outperform recent methods and achieves the best accuracy for 5 out of 12 classes and improves the mean class average by 1.3% w.r.t. to the previous SOTA result. To the best of our knowledge, this poses the best published result for the commonly used Adaptope dataset using a ResNet-50 backbone architecture. The domains are synthetic (S), real (R) and product (P).

Finally, we analyze the smoothness of the decision boundary. For this analysis, we employ the oscillation of classification metric described in Section 4.1. After the conclusion of CVP’s adaptation phase, we extract features μ for five instances per class and then calculate the sum of this metric for all distinct pairs (i, j), i.e. \( \sum_i \sum_j O(\mu(i), \mu(j)) \). The results are shown in Table 1 as mean and standard deviation over multiple runs. Applying CVP clearly leads to a much smoother decision boundary as quantified by the oscillation metric. For every analysed transfer task, the setup using CVP has a multiple standard deviations lower oscillation value, which is a direct indication of a smoother decision boundary. Overall, applying CVP results in an approximately 14 points lower oscillation metric, which translates to 14,000 fewer changes in classification. Smoothness of the decision boundary is also a strong indicator for the generalization capabilities of a model. This is especially important in UDA, as the trained model is employed as pseudo-labeler for subsequent training steps – a better generalization would necessarily lead to better pseudo-labels \( y_T(i) \). We therefore also provide domain adaptation results with and without CVP in Table 1. We observe that every model employing CVP improves in accuracy on the target domain when compared to a model without CVP. On average, CVP improves the domain adaptation results by 2.9%, with a 3.2% increase on the hardest transfer task S → R and up to 3.6% on S → P. We thus conclude that CVP can be seen as an intuitive uncertainty measure that improves domain adaptation capabilities by smoothing the decision boundary of a classifier. For further ablation studies, please refer to Appendix C.

Table 2: Per class accuracy (in %) for different methods on the VisDA-2017 test set as per challenge evaluation protocol. The domains are synthetic (source) and real (target).

| Method              | P→R | P→S | R→P | R→S | S→P | S→R | Avg. |
|---------------------|-----|-----|-----|-----|-----|-----|------|
| Source only         | 60.6±1.2 | 28.7±1.6 | 76.9±0.4 | 26.9±1.1 | 21.2±2.4 | 8.8±4.6 | 37.2 |
| RSDA-DANN [1, 2]    | 78.6±0.1 | 48.5±0.7 | 90.0±0.0 | 43.9±0.9 | 63.2±1.1 | 37.0±1.2 | 60.2 |
| RSDA-MSTN [3, 4]   | 73.8±0.2 | 59.2±0.2 | 87.5±0.2 | 50.3±0.5 | 69.5±0.6 | 44.6±1.1 | 64.2 |
| SymNets [5]         | 81.4±0.3 | 53.1±0.6 | 92.3±0.2 | 49.2±1.0 | 69.6±0.5 | 44.9±1.5 | 65.1 |
| CAN [6]             | 81.5±0.3 | 72.3±0.3 | 92.2±0.2 | 69.3±0.9 | 74.9±1.1 | 60.7±0.6 | 75.2 |
| CVP (ours)          | 81.7±0.1 | 71.6±0.4 | 91.5±0.1 | 71.8±0.8 | 77.2±1.7 | 62.3±1.3 | 76.0 |

Table 3: Classification accuracy (in %) for the Adaptope dataset using a ResNet-50 backbone architecture. The domains are synthetic (S), real (R) and product (P).
used ResNet-101 backbone. Given the current VisDA-2017 challenge leaderboard [23], our CVP method would rank 2nd place – only behind a much larger 5×ResNet-152 ensemble that also averages results over multiple runs with test time augmentations. This, however, is not a fair comparison to our single ResNet-101 setup without such test time tricks.

On the Adaptiope dataset (see Table 3), we compare to the recent CVPR proposals SymNets [47], CAN [21] and RSDA [14] (using both DANN [9] and MSTN [46]). Again, CVP is able to surpass these state-of-the-art algorithms and achieves the best accuracy on 4 out of 6 transfer tasks and also the overall average. Even for the hardest transfer tasks – i.e. those involving synthetic images as source domain – CVP manages to improve the results of CAN [21] by 1.6% for S→R and 2.3% for S→P. Again, CVP’s overall accuracy average of 76% poses the best published result and shows that our method can reliably conduct domain adaptation, even when confronted with Adaptiope’s 123 difficult classes.

Furthermore, we report results on the Office-Caltech dataset in Table 4. Again, our CVP method is able to achieve the best or second best results in 9 out of 12 transfer tasks and is also within 0.6% of UFAL [37]. Please note that due to Office-Caltech’s average target domain size of only 633 images, a 0.6% difference arises from less than 2 misclassifications per transfer task on average. Therefore, our CVP method can be considered on par with the top performing methods UFAL [37], PrDA [20] and RWA [44].

Finally, we report results on the Modern Office-31 dataset in Table 5. Yet again, CVP is able to achieve the best or second best result in 4 out of the 6 available transfer tasks and is on par with the recently proposed RSDA [14] and CAN [21] methods. For the difficult Aref→S task, CVP even manages to improve upon prior methods by 2.0%. We thus conclude that CVP is able to successfully conduct unsupervised domain adaptation on a wide variety of transfer tasks and datasets and is able to achieve results at or close to state-of-the-art level.

5 Conclusion

In this paper, we propose CVP – the Certainty Volume Prediction algorithm – for estimation of uncertainty in feature space. We show how CVP’s sampling stage interacts with the de-
cision boundary and how this can improve a model’s generalization properties and therefore domain adaptation capabilities. Additionally, we show that CVP’s $\sigma$ estimation in feature space relates to other common uncertainty quantifications in classifier space. Due to the intuitive interpretability of $\sigma$, we also see great potential for other downstream tasks, e.g. employing $\sigma$ for out-of-distribution rejection in open-set domain adaptation. Overall, we demonstrate that making a model uncertainty-aware through CVP can improve its predictions and achieve state-of-the-art results on challenging benchmark UDA datasets.

Appendices

A Implementation Details

In this section, we provide further detail about the employed setup and evaluation. For our experiments on VisDA-2017 [33], we employ the common ResNet-101 [17] feature extractor ($f_{CNN}$) similar to [6, 21, 37], while for Office-Caltech [11], Modern Office-31 [38] and Adaptiope [38], we use ResNet-50 as proposed in [38]. In all cases, $f_{CNN}$ is pretrained on ImageNet [5]. We implement the $f_\Sigma$ subnetwork as a two layer linear network. The input size is 2048 (same size as $\mu$), hidden size 2048 and output size 1. Intermediate outputs are activated by ReLU. The final output is activated by the softplus function in order to enforce $\sigma \in \mathbb{R}^+_{0}$ and the positive semi-definiteness of $\Sigma$.

We jointly optimize all parameters using SGD with Nesterov momentum [30] of 0.95 and a learning rate of $5 \times 10^{-4}$, which is decreased according to the schedule used in [21]. Networks are trained for 250 cycles with $T_{cycle} = 50$ forward passes each. For the evaluation of the decision boundary, we set $K = 1000$ in order to sample 1000 subfeatures (including start and end feature). Our code is implemented in PyTorch [32] and will be made available.

Reparameterization Trick. Neural network training is based on the idea of backpropagation and therefore requires the whole network to be differentiable. This, however, is not the case when including a generally non-differentiable step such as sampling from a distribution. In this work, we want the parameters of $f_\Sigma$ to also be updated based on our proposed $\mathcal{L}_{ce}^\phi$ loss function. We thus sidestep the non-differentiability of sampling by applying the reparameterization trick proposed in [24]. Here, instead of sampling a feature $z$ directly from $z \sim \mathcal{N}(\mu, \Sigma)$, we first draw an auxiliary noise variable $\epsilon \sim \mathcal{N}(0, 1)$ and then compute $z = \mu + \epsilon \Sigma^{\frac{1}{2}}$. This can also be rewritten as $\forall i: z_i = \mu_i + \epsilon_i \text{diag}(\Sigma^{\frac{1}{2}})_i$ with $\epsilon_i \sim \mathcal{N}(0, 1)$ (note the scalar notation). Applying this reparameterization allows the gradient to also flow through our $f_\Sigma$ subnetwork.

B Theorem for $\kappa$ Initialization

We now derive the constant $\kappa$ (used in the $\psi^{(i)}$ equation in the main paper) as the expected cross-entropy value given a randomly initialized classifier. Consider the cross-entropy term $H(q, p) = -\sum q \log(p)$ between two categorical distributions $q$ and $p$ with $|C|$ classes. Given an uniform class distribution, the one-hot encoded target $q_i$ will equal 1 at position $i$ with probability $\frac{1}{|C|}$. We can then rewrite $H(q, p) = -\sum_{i=1}^{\lfloor |C| / 2 \rfloor} \frac{1}{|C|} \log(p_i)$. This can be further simplified as Equation 4.
\[ H(q, p) = - \frac{1}{|C|} \sum_{i=1}^{|C|} \log(p_i) = - \frac{1}{|C|} \left( \log(p_1) + \ldots + \log(p_{|C|}) \right) = - \frac{1}{|C|} \log(p_1 \cdot p_2 \ldots p_{|C|}) \]

(4)

We can assume that the worst state of a model during training is to be equal to a random oracle, i.e. we assume that \( \forall i : \mathbb{E}[p_i] \approx \frac{1}{|C|} \) holds true. Given this assumption, we can rewrite Equation 4 as

\[ H(q, p) = - \frac{1}{|C|} \log(\frac{1}{|C|}) = - |C| \log(\frac{1}{|C|}) = - \log(\frac{1}{|C|}) \]

which further simplifies to \( \log(|C|) \). We thus define constant \( \kappa = \log(|C|) \), which can be seen as an expected upper bound for \( L^\phi_{ce} \).

C Additional Analysis

Ablation Study. We also evaluate our proposed CVP method by conducting an ablation study w.r.t. its two hyperparameters \( \alpha \) and \( M \). For this evaluation, we use Adaptope’s S→P and S→R transfer tasks. Hyperparameter \( M \) controls the amount of samples that are drawn from the distribution predicted by CVP. We analyze \( M \) in the range \([2^5, 2^8]\) and plot the results in Figure Ia. We notice that CVP is fairly robust w.r.t. to this parameter as long as enough samples are drawn. Overall \( M = 2^6 \) resulted in the best performance. Similarly, we analyze the samples weight \( \alpha \) in Figure Ib. We observe that choosing \( \alpha \) either too low or too high degrades the accuracy consistently for both transfer tasks. Intuitively, a too high \( \alpha \) would attribute the generated samples the same weight as the predicted mean feature \( \mu \), which can lead to optimization problems, as the samples are randomly drawn and not an actual output of the feature extractor. Related to this, setting \( \alpha \) too low removes the additional supervision to enforce the cluster assumption. We thus set \( \alpha = 0.5 \) for our experiments, due to its overall best performance.

Sigma Analysis. Furthermore, we provide an additional analysis for the P→S task of Adaptope similar to the analysis in the main paper in Figure Ic. We note the same key observations as for the S→P and S→R tasks in the main paper: Directly after the switch from the pretraining to the adaptation phase, \( \sigma \) drops harshly due to being confronted with new, unseen target domain data. For the source domain, \( \sigma \) recovers quickly while for the target domain, \( \sigma \) grows slower due to the ongoing domain adaptation. Unlike the S→* tasks, the source and target domain \( \sigma \) end up very close to each other for the S→P task. This, however, is expected due to the smaller domain gap and the missing background clutter: both the synthetic and the product domain have cropped items with either a fully white (product domain) or fully grey (synthetic domain) background.

Interpretability of \( \sigma \). We further analyze the interpretability of \( \sigma \). For this, we run our full CVP method on Adaptope’s S→R task and extract \( \sigma \) values for the whole target domain. In Figure II, we show selected results from the top and bottom 20 \( \sigma \) values. Images associated with high magnitudes of \( \sigma \) are usually taken under good conditions w.r.t. lighting and camera angle and commonly represent the best images of their respective object classes. For low magnitudes of \( \sigma \), we often find objects to be in unusual positions (see Figure IIa), disassembled or broken (see Figure IIa) or in exceptionally uncommon conditions or situations (see Figure IIe). This further reinforces the usability of \( \sigma \) as an interpretable uncertainty measure.
Figure I: (a, b) Results for different ablation studies. (c) Median $\sigma$ values for the source and target domain as the training progresses. Best viewed in the digital version.

(a) Number of Samples $M$  (b) Samples Weight $\alpha$  (c) P→S

Figure II: Left two columns: Images associated with a low magnitude of $\sigma$. Right two columns: Images associated with high magnitudes of $\sigma$. Image captions denote the ground truth class.

(a) axe  (b) acoustic guitar  (c) hat  (d) pipe wrench  (e) skeleton  (f) office chair  (g) tent

Low $\sigma$ magnitude  High $\sigma$ magnitude
In Figure III, we provide a feature visualization for the features learned by our CVP method. For this, we extract the 2048-dimensional feature $\mu$ for every target domain image in Adaptope’s $S \rightarrow P$ and $S \rightarrow R$ transfer tasks respectively. The feature extraction step is conducted right after the source-only pretraining and after the finalized domain adaptation phase. We then employ t-SNE [27] in order to project the features into a two-dimensional space. As evident from Figure IIIa, the features show no discernible structure with regard to clusters or proximity of same-class features after the source-only pretraining step. However, after the adaptation phase with CVP (Figure IIIb), clear clusters of same-class features can be observed. We thus conclude that – even for the very hard synthetic to real and synthetic to product transfer tasks of Adaptope – our CVP method was able to learn discriminative features even though no target domain labels were available.

![Figure III: Visualization of target domain features after the source-only pretraining and after the domain adaptation with CVP. Features were extracted for Adaptope’s $S \rightarrow P$ (top row) and $S \rightarrow R$ (bottom row) transfer tasks. Every color represents one of Adaptope’s 123 classes. Please note that due to the high number of classes, some colors are hard to distinguish.](image_url)
References

[1] Amir Atapour-Abarghouei and Toby P Breckon. Real-time monocular depth estimation using synthetic data with domain adaptation via image style transfer. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pages 2800–2810, 2018.

[2] Peter Bartlett and John Shawe-taylor. Generalization performance of support vector machines and other pattern classifiers, 1998.

[3] Woong-Gi Chang, Tackgeun You, Seonguk Seo, Suha Kwak, and Bohyung Han. Domain-specific batch normalization for unsupervised domain adaptation. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pages 7354–7362, 2019.

[4] Olivier Chapelle and Alexander Zien. Semi-supervised classification by low density separation. In AISTATS, volume 2005, pages 57–64. Citeseer, 2005.

[5] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In 2009 IEEE conference on computer vision and pattern recognition, pages 248–255. Ieee, 2009.

[6] Geoff French, Michal Mackiewicz, and Mark Fisher. Self-ensembling for visual domain adaptation. In International Conference on Learning Representations, 2018. URL https://openreview.net/forum?id=rkpoTaxA-.

[7] B. Chidlovskii G. Csurka and S. Clinchant. VisDA Classification Challenge:Runner-Up Talk. https://ai.bu.edu/visda-2017/assets/attachments/VisDA_NaverLabs.pdf, 2020. Accessed: 2020-03-31.

[8] Yarin Gal and Zoubin Ghahramani. Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In international conference on machine learning, pages 1050–1059, 2016.

[9] Yaroslav Ganin and Victor Lempitsky. Unsupervised domain adaptation by backpropagation. In Proceedings of the 32Nd International Conference on International Conference on Machine Learning - Volume 37, ICML’15, pages 1180–1189. JMLR.org, 2015. URL http://dl.acm.org/citation.cfm?id=3045118.3045244.

[10] Behnam Gholami, Vladimir Pavlovic, et al. Punda: Probabilistic unsupervised domain adaptation for knowledge transfer across visual categories. In Proceedings of the IEEE International Conference on Computer Vision, pages 3581–3590, 2017.

[11] Boqing Gong, Yuan Shi, Fei Sha, and Kristen Grauman. Geodesic flow kernel for unsupervised domain adaptation. In 2012 IEEE Conference on Computer Vision and Pattern Recognition, pages 2066–2073. IEEE, 2012.

[12] Ian J Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In Advances in neural information processing systems, page 2672–2680, 2014.
[13] Arthur Gretton, Karsten Borgwardt, Malte Rasch, Bernhard Schölkopf, and Alex Smola. A kernel method for the two-sample problem. *Advances in neural information processing systems*, 19:513–520, 2006.

[14] Xiang Gu, Jian Sun, and Zongben Xu. Spherical space domain adaptation with robust pseudo-label loss. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9101–9110, 2020.

[15] Dayan Guan, Jiaxing Huang, Aoran Xiao, Shijian Lu, and Yanpeng Cao. Uncertainty-aware unsupervised domain adaptation in object detection. *IEEE Transactions on Multimedia*, 2021.

[16] Ligong Han, Yang Zou, Ruijijiang Gao, Lezi Wang, and Dimitris Metaxas. Unsupervised domain adaptation via calibrating uncertainties. In *CVPR Workshops*, volume 9, 2019.

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

[18] K. E. Hindi and M. Al-Akhras. Smoothing decision boundaries to avoid overfitting in neural network training. *Neural Network World*, 21:311–325, 2011.

[19] 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 *International Conference on Machine Learning (ICML)*, 2018.

[20] Kevin Hua and Yuhong Guo. Unsupervised domain adaptation with progressive domain augmentation. *arXiv preprint arXiv:2004.01735*, 2020.

[21] Guoliang Kang, Lu Jiang, Yi Yang, and Alexander G Hauptmann. Contrastive adaptation network for unsupervised domain adaptation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4893–4902, 2019.

[22] Hamid Karimi, Tyler Derr, and Jiliang Tang. Characterizing the decision boundary of deep neural networks. *arXiv preprint arXiv:1912.11460*, 2019.

[23] Ben Usman Kate Saenko and Xingchao Peng. 2017 Visual Domain Adaptation (VisDA2017) Classification Challenge. [https://competitions.codalab.org/competitions/17052](https://competitions.codalab.org/competitions/17052), 2017. Accessed: 2021-04-06.

[24] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

[25] Wee Sun Lee, Peter L Bartlett, and Robert C Williamson. Lower bounds on the vc dimension of smoothly parameterized function classes. *Neural Computation*, 7(5): 1040–1053, 1995.

[26] Mingsheng Long, Jianmin Wang, Guiguang Ding, Jiaguang Sun, and Philip S Yu. Transfer feature learning with joint distribution adaptation. In *Proceedings of the IEEE international conference on computer vision*, pages 2200–2207, 2013.

[27] Laurens van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of machine learning research*, 9(Nov):2579–2605, 2008.
[28] Zhong Meng, Jinyu Li, Yifan Gong, and Biing-Hwang Juang. Adversarial teacher-student learning for unsupervised domain adaptation. In 2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), pages 5949–5953. IEEE, 2018.

[29] Zak Murez, Soheil Kolouri, David Kriegman, Ravi Ramamoorthi, and Kyungnam Kim. Image to image translation for domain adaptation. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pages 4500–4509, 2018.

[30] Yurii E Nesterov. A method for solving the convex programming problem with convergence rate $o(1/k^2)$. In Dokl. akad. nauk Sssr, volume 269, pages 543–547, 1983.

[31] Sungrae Park, JunKeon Park, Su-Jin Shin, and Il-Chul Moon. Adversarial dropout for supervised and semi-supervised learning. In Thirty-Second AAAI Conference on Artificial Intelligence, 2018.

[32] Adam Paszke, Sam Gross, Soumith Chintala, et al. Automatic differentiation in PyTorch. In NIPS Autodiff Workshop, 2017.

[33] Xingchao Peng, Ben Usman, Neela Kaushik, Judy Hoffman, Dequan Wang, and Kate Saenko. Visda: The visual domain adaptation challenge, 2017.

[34] Pedro O Pinheiro. Unsupervised domain adaptation with similarity learning. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pages 8004–8013, 2018.

[35] Esteban Real, Jonathon Shlens, Stefano Mazzocchi, Xin Pan, and Vincent Vanhoucke. Youtube-boundingboxes: A large high-precision human-annotated data set for object detection in video. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pages 5296–5305, 2017.

[36] Stephan R Richter, Vibhav Vineet, et al. Playing for data: Ground truth from computer games. In European conference on computer vision, pages 102–118. Springer, 2016.

[37] Tobias Ringwald and Rainer Stiefelhagen. Unsupervised Domain Adaptation by Uncertain Feature Alignment. In The British Machine Vision Conference (BMVC), 2020.

[38] Tobias Ringwald and Rainer Stiefelhagen. Adaptiope: A modern benchmark for unsupervised domain adaptation. In Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision, pages 101–110, 2021.

[39] Pau Rodríguez, Issam Laradji, Alexandre Drouin, and Alexandre Lacoste. Embedding propagation: Smoother manifold for few-shot classification. In European Conference on Computer Vision, pages 121–138. Springer, 2020.

[40] Kate Saenko, Brian Kulis, Mario Fritz, and Trevor Darrell. Adapting visual category models to new domains. In European conference on computer vision, pages 213–226. Springer, 2010.

[41] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. The journal of machine learning research, 15(1):1929–1958, 2014.
[42] Baochen Sun, Jiashi Feng, and Kate Saenko. Return of frustratingly easy domain adaptation. In Proceedings of the AAAI Conference on Artificial Intelligence, volume 30, 2016.

[43] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. Intriguing properties of neural networks. arXiv preprint arXiv:1312.6199, 2013.

[44] Twan van Laarhoven and Elena Marchiori. Unsupervised domain adaptation with random walks on target labelings, 2017.

[45] Sebastiano Vascon, Sinem Aslan, Alessandro Torcinovich, Twan van Laarhoven, Elena Marchiori, and Marcello Pelillo. Unsupervised domain adaptation using graph transduction games. In 2019 International Joint Conference on Neural Networks (IJCNN), pages 1–8. IEEE, 2019.

[46] Shaoan Xie, Zibin Zheng, Liang Chen, and Chuan Chen. Learning semantic representations for unsupervised domain adaptation. In International Conference on Machine Learning, pages 5419–5428, 2018.

[47] Yabin Zhang, Hui Tang, Kui Jia, and Mingkui Tan. Domain-symmetric networks for adversarial domain adaptation. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pages 5031–5040, 2019.

[48] Zhengxia Zou, Tianyang Shi, Wenyuan Li, Zhou Zhang, and Zhenwei Shi. Do game data generalize well for remote sensing image segmentation? Remote Sensing, 12(2):275, 2020.