Unsupervised Domain Adaptation via Calibrating Uncertainties

Ligong Han¹, Yang Zou², Ruijiang Gao³, Lezi Wang¹, and Dimitris Metaxas¹

¹Department of Computer Science, Rutgers University
²Department of Electrical and Computer Engineering, Carnegie Mellon University
³McCombs School of Business, The University of Texas at Austin

l.han@rutgers.edu  yzou2@andrew.cmu.edu  ruijiang@utexas.edu  lw462@cs.rutgers.edu  dnm@cs.rutgers.edu

Abstract

Unsupervised domain adaptation (UDA) aims at inferring class labels for unlabeled target domain given a related labeled source dataset. Intuitively, the model trained on labeled data will produce high uncertainty estimation for unseen data. Under this assumption, models trained in the source domain would produce high uncertainties when tested on the target domain. In this work, we build on this assumption and propose to adapt from source and target domain via calibrating their predictive uncertainties. We employ variational Bayes learning for uncertainty estimation which is quantified as the predicted Rényi entropy on the target domain. We discuss the theoretical properties of our proposed framework and demonstrate its effectiveness on three domain-adaptation tasks.

1. Introduction

The ability to model uncertainty is important in unsupervised domain adaptation (UDA). For example, self-training-based approaches [13, 27] often requires the model to reliably estimate the uncertainty of its prediction on target domain in the pseudo-label selection phase. However, traditional deep neural networks (DNN) can easily assign high confidence to a wrong prediction [4, 15], thus are not able to reliably and quantitatively render the uncertainty given data.

Bayesian Neural Networks (BNN) [17, 4, 1, 9] tackles this problem by taking a Bayesian view of the training process. Instead of obtaining a point estimate of weights, BNN tries to model the distributions over weights. We leverage BNN as a powerful tool to model uncertainties and the investigation on the uncertainties among different domains provides us insights on addressing domain adaptation problem. An observation is that a BNN trained on source domain would produce much higher uncertainties when deployed on target domain. Our uncertainty-based domain-adaptation approach is built on the intuition that a model gives similar uncertainty estimations on the two domains learns to adapt from source to target well. Thus, we propose to directly match the estimated uncertainty between source and target domain during training.

Our contributions are listed as follows:

- We propose a novel framework for unsupervised domain adaptation by calibrating the predictive uncertainty.
- We adopt variational Bayes neural network for uncertainty estimation and discuss its relationship with entropy regularization [6] and self-training [13].
- Preliminary results show that the proposed BNN-based uncertainty calibration is effective and stable in training.

2. Related Work

Shannon entropy is commonly used to quantify the uncertainty of a given distribution. Entropy-based UDA has already been proposed in [23]. Unlike [23], we avoid using adversarial learning which tends to be unstable and hard to train. Also, entropy regularization is proposed in [6] for semi-supervised learning and can be directly applied to UDA. However, our framework is more general since the uncertainty is not necessarily to be the Shannon entropy. In fact, we formalize the uncertainty as Rényi entropy which is a generalization of Shannon entropy. Many other methods in UDA can be modeled under this framework, for example, self-train [13, 27] can be viewed as minimizing the min-entropy which is a special case of Rényi entropy.

As pointed out by [5], directly optimizing the estimated Shannon entropy given data requires the classifier to be locally-Lipschitz [16]. Co-DA [11] and DIRT-T [21] propose to solve this problem by incorporate the
locally-Lipschitz constraint via virtual adversarial training (VAT) [16].

Another complimentary line of research employs self-ensemble and shows promising results [2]. Indeed, BNN [4] performs Bayesian ensembling by nature. This is part of the reason why BNN provides a better uncertainty estimation.

3. Uncertainty in Deep Neural Networks

BNN models the uncertainty in DNNs by estimating a posterior over the network parameters. Given the dataset \( D = \{ (x(i), y(i)) \}^{N}_{i=1} \), the output of BNN is denoted as \( f(x|w) \) where \( x \) is input data and \( w \) is the weights (or parameters). For classification task, \( f \) is the predicted logits and the resulting probability vector is given by a softmax function: \( P(y|x, w) = \text{softmax}(f(x|w)) \). The predictive distribution over labels given input \( x \) is: \( P(y|x) = \mathbb{E}_{P(w|D)}P(y|x, w) \). Thus, the predictive uncertainty can be quantified as the Rényi entropy, \( H_\alpha(P(y|x)) \). Rényi entropy [24] of order \( \alpha (\alpha > 0) \) is defined as

\[
H_\alpha(P) = \frac{1}{1 - \alpha} \log(\sum_k (P_k)^\alpha). \tag{1}
\]

Plugging the above equation into the MLE term in ELBO, the BNN is trained via a cross-entropy (CE) loss plus weight decay.

4. Domain Adaptation via Calibrating Uncertainties

Denote source and target dataset as \( D_S = \{ (x^{(s)}, y^{(s)}) \}^{s}_{s \in S} \) and \( D_T = \{ (x^{(t)}) \}^{t}_{t \in T} \) respectively, where \( x^{(s)}, x^{(t)} \) indicate the samples and \( y^{(s)} \) is the label in source domain, and \( D = D_S \cup D_T \). We propose to calibrate the predictive uncertainty of target dataset with the source domain uncertainties. Concretely, we minimize the cross-entropy loss in the source domain with the constraint of the predicted entropy (uncertainty) in the target domain:

\[
\min_\theta \mathcal{L}_{CE} = \frac{1}{|S|} \sum_{s \in S} H_{CE}(y^{(s)}, P(y|x^{(s)}; \theta)) \nonumber
\]

s.t. \( \frac{1}{|T|} \sum_{t \in T} H_\alpha(P(y|x^{(t)}; \theta)) \leq C, \tag{4} \)

where \( H_{CE}(\cdot, \cdot) \) is the cross-entropy and \( C \) indicates the strength of the applied constraint. Rewriting Eq. 4 as a Lagrangian with a multiplier \( \beta \),

\[
\mathcal{F} = \frac{1}{|S|} \sum_{s \in S} H_{CE}(y^{(s)}, P(y|x^{(s)}; \theta)) + \beta \left( \frac{1}{|T|} \sum_{t \in T} H_\alpha(P(y|x^{(t)}; \theta)) - C \right). \tag{5} \]

Since \( \beta, C \geq 0 \) an upper bound on \( \mathcal{F} \) is obtained,

\[
\mathcal{F} \leq \frac{1}{|S|} \sum_{s \in S} H_{CE}(y^{(s)}, P(y|x^{(s)}; \theta)) + \beta \left( \frac{1}{|T|} \sum_{t \in T} H_\alpha(P(y|x^{(t)}; \theta)) = \mathcal{L}_\alpha. \tag{6} \]

In theory, Eq. 5 can be optimized via dual gradient descent and \( \beta \) is jointly updated along with \( \theta \). For simplicity, we follow the work of [8] and fix \( \beta \) as a hyper-parameter in the experiment and minimize the upper bound \( \mathcal{L}_\alpha \).

Note that letting \( \alpha \rightarrow 1 \) in Eq. 6 is in fact the (Shannon) entropy regularization as described in [5, 6], except that here we consider a variational BNN. As pointed out in [6], directly optimizing Eq. 6 can be difficult and expectation maximization (EM) algorithms are often used. Proposed in [25, 6], deterministic annealing EM anneals the predicted probabilities as soft-labels and minimizes the resulting cross-entropy. In the extreme case, soft-labels become one-hot vectors and the algorithm turns out to be self-training with pseudo-labels [13]. In our Rényi entropy regularization framework, self-training is essentially optimizing
the min-entropy ($\alpha \rightarrow \infty$). Then the objective reads

$$
L_{\infty} = \frac{1}{|S|} \sum_{s \in S} H_{CE}(y^{(s)}, P(y|x^{(s)}; \theta)) + \frac{\beta}{|T|} \sum_{t \in T} H_{CE}(\hat{y}^{(t)}, P(y|x^{(t)}; \theta)),
$$

(7)

with $\hat{y}^{(t)} = \text{onehot}({\arg \max}_{k \in \{1, ..., K\}} P(y_k|x^{(t)}; \theta))$ to be pseudo-labels in target domain. Subscript $k$ denotes the $k$-th element in a given $K$-dim vector. The relationship between $L_1$ and $L_{\infty}$ can be immediately realized by noticing that the Shannon entropy is an upper bound of the min-entropy:

$$
H_1(P) = -\sum_k P_k \log(P_k) \geq -\sum_k P_k \log(\max_k P_k) = -\log(\max_k P_k) = H_{\infty}(P) = H_{CE}(\hat{y}, P)
$$

(8)

We build our method on top of class-balanced self-training (CBST) proposed in [27]. CBST seeks to generate pseudo-labels from the most confident predictions that follows an "easy-to-hard" scheme, since jointly learning the model and optimizing pseudo-labels on all unlabeled data is naturally difficult. The authors also propose to normalize the class-wise confidence levels in pseudo-label generation to balance the class distribution. For a detailed formulation, we suggest readers referring Section 4.1 and 4.2 in [27].

5. Experiments

We first show results on three toy datasets MNIST [12], USPS and SVHN [18], where we consider **MNIST→USPS** and **SVHN→MNIST**. Then we present preliminary results on a challenging benchmark: **VisDA17** (classification) [19] which contains 12 classes. We follow the standard protocol in [19, 22, 20].

The accuracies on source and target domains for base models are reported in Table 1. We use DTN [26] as our base model for MNIST→USPS and SVHN→MNIST. To implement its Bayesian variant (BDTN), we add another classifier to predict the logarithm of variance.

Domain adaptation results are shown in Table 2. We can see self-training with pseudo-labels (CBST-BNN-∞) are more stable than directly minimizing the predicted Shannon entropy (CBST-BNN-1). Mean accuracies on VisDA17 dataset are reported in Table 3. Following the protocol in [27], we train a standard ResNet101 [7] as the base model and add a second classifier (denoted as BRes101) to predict logarithm of variance on logits.

6. Conclusion

In this work, we propose to calibrate the predictive uncertainty for unsupervised domain adaptation. The uncertainty is quantified via Bayesian networks under a general Rényi entropy regularization framework. Results show the uncertainty estimation by Bayesian networks is effective and leads to stable performance in unsupervised domain adaptation.

### Table 1: Training base models on MNIST and SVHN.

<table>
<thead>
<tr>
<th>Model</th>
<th>Source Acc</th>
<th>Target Acc</th>
</tr>
</thead>
<tbody>
<tr>
<td>DTN</td>
<td>76.81 ± 1.39</td>
<td>80.59 ± 34.56</td>
</tr>
<tr>
<td>BDTN-M10</td>
<td>99.42 ± 7.38</td>
<td>99.42 ± 7.38</td>
</tr>
<tr>
<td>BDTN-M20</td>
<td>100.00 ± 0.00</td>
<td>99.42 ± 7.38</td>
</tr>
<tr>
<td>BDTN-M100</td>
<td>100.00 ± 0.00</td>
<td>99.42 ± 7.38</td>
</tr>
</tbody>
</table>

**Table 1:** Training base models on MNIST and SVHN. BDTN is a modified Bayesian DTN [26], with different $M$ values.

### Table 2: Results on MNIST→USPS and SVHN→MNIST.

<table>
<thead>
<tr>
<th>Model</th>
<th>Target Acc</th>
<th>Acc Gain</th>
</tr>
</thead>
<tbody>
<tr>
<td>Source-DTN</td>
<td>83.94 ± 34.56</td>
<td>-</td>
</tr>
<tr>
<td>Source-BDTN</td>
<td>84.78 ± 34.56</td>
<td>-</td>
</tr>
<tr>
<td>CBST</td>
<td>93.20 ± 9.26</td>
<td>9.26</td>
</tr>
<tr>
<td>CBST-BNN-1</td>
<td>89.31 ± 4.53</td>
<td>4.53</td>
</tr>
<tr>
<td>CBST-BNN-∞</td>
<td>93.85 ± 9.07</td>
<td>9.07</td>
</tr>
</tbody>
</table>

**Table 2:** Results on MNIST→USPS and SVHN→MNIST. CBST [27] uses DTN as the base model for self-training. CBST-BNN-∞ uses BDTN as the base model and optimizes $L_{\infty}$, while CBST-BNN-1 optimizes $L_1$.

### Table 3: Preliminary results on VisDA17 [19] classification benchmark.

<table>
<thead>
<tr>
<th>Model</th>
<th>Target Acc</th>
<th>Acc Gain</th>
</tr>
</thead>
<tbody>
<tr>
<td>Source-Res101</td>
<td>48.02 ± 28.79</td>
<td>-</td>
</tr>
<tr>
<td>Source-BRes101</td>
<td>46.03 ± 28.79</td>
<td>-</td>
</tr>
<tr>
<td>CBST</td>
<td>76.81 ± 28.79</td>
<td>28.79</td>
</tr>
<tr>
<td>CBST-BNN-∞</td>
<td>80.59 ± 34.56</td>
<td>34.56</td>
</tr>
</tbody>
</table>

**Table 3:** Preliminary results on VisDA17 [19] classification benchmark.
References