Independent Feature Decomposition and Instance Alignment for Unsupervised Domain Adaptation

Qichen He\textsuperscript{1}, Siying Xiao\textsuperscript{1}, Mao Ye\textsuperscript{1}\textsuperscript{*}, Xiatan Zhu\textsuperscript{2}, Ferrante Neri\textsuperscript{3} and Dongde Hou\textsuperscript{4}

\textsuperscript{1}School of CSE, University of Electronic Science and Technology of China, Chengdu, China
\textsuperscript{2}Surrey Institute for People-Centred Artificial Intelligence, CVSSP, University of Surrey, Guildford, UK
\textsuperscript{3}NICE Research Group, Department of Computer Science, University of Surrey, Guildford, UK
\textsuperscript{4}Advanced Research Institute, Southwest University of Political Science&Law, Chongqing, China
\{202121081209, 2018270101016\}@std.uestc.edu.cn, maoye@uestc.edu.cn, \{xiatian.zhu, f.neri\}@surrey.ac.uk, gxin_001@163.com

Abstract

Existing Unsupervised Domain Adaptation (UDA) methods typically attempt to perform knowledge transfer in a domain-invariant space explicitly or implicitly. In practice, however, the obtained features are often mixed with domain-specific information which causes performance degradation. To overcome this fundamental limitation, this article presents a novel independent feature decomposition and instance alignment method (IndUDA in short). Specifically, based on an invertible flow, we project the base feature into a decomposed latent space with domain-invariant and domain-specific dimensions. To drive semantic decomposition independently, we then swap the domain-invariant part across source and target domain samples with the same category and require their inverted features are consistent in class-level with the original features. By treating domain-specific information as noise, we replace it by Gaussian noise and further regularize source model training by instance alignment, i.e., requiring the base features close to the corresponding reconstructed features, respectively. Extensive experiment results demonstrate that our method achieves state-of-the-art performance on popular UDA benchmarks. The appendix and code are available at https://github.com/ayombeach/IndUDA.

1 Introduction

Recently, deep learning has achieved great success in a variety of applications, particularly in computer vision and natural language processing. Despite such a great achievement, most deep learning methods require a large amount of manually labelled data. However, in many real-world applications, such an amount of data with labels is usually hard to collect [Pan and Yang, 2009]. In the majority of cases, we can only have access to a model pre-trained on a labelled source domain and fit it into an unlabelled target domain [Long et al., 2016; Pan et al., 2010]. Due to domain shift, the source domain model often suffers from performance degradation on target data. To solve this problem, the research of Unsupervised Domain Adaptation (UDA) comes into play [Sugiyama et al., 2007]. Usually, UDA methods project the source and target samples into a common feature space to enable knowledge transfer cross domains. This feature space is often called a domain-invariant space. Existing methods typically derive this domain-invariant space by three categories of approaches. The first category is based on traditional distribution alignment [Long et al., 2015; Long et al., 2016; Ganin et al., 2016], which minimizes domain discrepancy by aligning the distributions between two domains. Domain-invariant features are extracted without considering the impact of domain-specific information (playing a role as noise as they are largely class irrelatively). The second approach is self-supervised learning which implicitly learns a common space based on pseudo labels or prototypes [Saenko et al., 2010; French et al., 2018]. Similar to the first category of methods, the impact of domain-specific noise is not

\textsuperscript{*}Mao Ye is the corresponding author. This work was supported by National Natural Science Foundation of China (62276048).
considered. Hence, the ideal domain-invariant space cannot be achieved. The last category disentangles the feature into domain-invariant feature and domain-specific feature [Cai et al., 2019; Lee et al., 2021; Zhou et al., 2022a]. Although good performance can be achieved, the independence relationship between the disentangled domain-invariant feature and domain-specific feature is largely overlooked.

In summary, previous approaches presented in the literature can only obtain a noisy domain-invariant space, as illustrated in Fig.1(a). To address this limitation, we propose to decompose a sample feature into two independent parts in channel: a domain-invariant part and a domain-specific part. It is not so easy since the extracted features are dependent in channels. To that end, we exploit normalizing flow [Lugmayr et al., 2020] that can transfer a distribution to another distribution while holds an important invertible property. As shown in Fig.1(b), our key idea is that, through swapping domain-invariant parts between a source sample and a target sample with the same category in the latent space whilst requiring the inverted features after swapping are consistent with the original feature in class-level, the independence between domain-invariant and domain-specific information can be achieved to high degree. Then, if we discard the domain-specific part, and require the features from the source model close to the corresponding inverted features, the model is endowed with the ability of extracting domain invariant features.

With the above analysis, we propose a novel method, dubbed as Independent feature decomposition and instance alignment UDA (IndUDA). Specifically, IndUDA consists of an independent feature decomposition module and an instance feature alignment module. For the first module, an Invertible Neural Network (INN) [Lugmayr et al., 2020] is exploited to project the original feature into a compact latent space. A channel mask is further used to split the projected feature into domain-invariant part and domain-specific part. After that, for the target samples with high-confidence pseudo-labels, the instance cross-domain swap strategy exchanges the domain-invariant parts with respect to the source and target samples, then requires consistency in class-level between the original feature and inverted feature through a backward process of INN flow. For the instance feature alignment module, the domain-specific part is dropped in latent space and replaced by Gaussian noise. An ideal domain-invariant feature is obtained through the reverse process of INN flow again. Then, the source and target features are aligned with the ideal features respectively. In this way, domain-invariant parts in the original space can be obtained.

Our contribution can be summarized as follows: (1) We propose a novel unsupervised domain adaptation framework, namely IndUDA. Different from previous approaches, the domain-specific noise can be better eliminated by requiring independence between domain-invariant parts and domain-specific part. So the performance is boosted. (2) A new instance-level cross-domain feature swap strategy is proposed to ensure independence, along with exploiting the invertible flow. By swapping the domain-invariant parts between the source and target sample features and requiring the inverted features consistent, the independence degree between domain-invariant part to domain-specific part can be maximized. (3) Extensive experiments demonstrate the efficacy of our method on three benchmarks. Ablation analysis explains the performance superiority of our model in the component level. Furthermore, the adaptation networks are only used in the training phase, while during the test process, only the adapted model is needed.

2 Related Work

2.1 Unsupervised Domain Adaptation

Unsupervised domain adaptation (UDA) is a technique that aims to improve the generalization ability of a model on an unlabelled target domain [Pan et al., 2010; Patel et al., 2015; Tang et al., 2020]. From the point of finding domain-invariant features, the existing solutions can be roughly divided into three categories. The first approach is based on self-supervised learning, such as Self-Ensembling (SE) [French et al., 2018] and Transferrable Prototypical Networks (TPN) [Pan et al., 2019], which attempt to apply pseudo-labels or other supervision information like a prototype to retrain the source model. Such a self-training solution is also a way to find a common representation space of two domains [Saenko et al., 2010].

The second approach regards domain adaptation as a distribution alignment problem. There are two routes to address this problem: statistical moment matching and adversarial learning. The statistical moment matching method aims to minimize the statistical discrepancy to align the distributions between two domains, such as Maximum Mean Discrepancy (MMD) [Long et al., 2015], DAN [Long et al., 2015], and CAN [Kang et al., 2019]. Adversarial learning-based methods use a min-max training approach between feature extractor and discriminator to find domain-invariant features, such as MCD [Saito et al., 2018], DANN [Ganin and Lempitsky, 2015] and ADDA [Tzeng et al., 2017].

The final approach aims to disentangle features into domain-invariant and domain-specific features [Cai et al., 2019; Lee et al., 2021; Liu et al., 2018; Deng et al., 2022]. There are two ways. The first is based on a classifier prototype. For example, MSDTR [Zhou et al., 2022a] uses classifier prototypes and domain prototypes to split features. Another one trains an encoder-decoder network to decompose and reconstruct features. For example, DRANet [Lee et al., 2021] adds a content adaptive domain transfer module in an encoder-decoder network that helps to retain scene structure. Despite the great performance achieved, all of these methods do not systematically consider independence between domain-invariant and domain-specific features. So domain-specific noise is not eliminated very well.

2.2 Normalizing Flow

Normalized flow was originally designed for unsupervised learning of probabilistic models [Dinh et al., 2016]. The flow model regards the origin dataset as a complex distribution with an invertible neural network, which can map data sampled from a simple distribution (e.g. Gaussian). Previous methods explore the exact negative log-likelihood based on conditional distribution [Kingma and Dhariwal, 2018] which
forces the simple distribution to approximate the prior distribution. A series of works exploit the flow model on many low-level visual tasks such as InvDN [Liu et al., 2021b] for image denoising tasks and SRFlow [Lugmayr et al., 2020] for super-resolution. Different from previous works, we employ normalizing flow to implement the independence between the domain-invariant part and domain-specific part by sufficiently using the invertible property.

3 Methodology

3.1 Problem Statement

In the UDA setting, we have a set of labelled source domain samples $S = \{(x^s_1, y^s_1), \cdots, (x^s_{N_s}, y^s_{N_s})\}$ and an unlabelled target domain sample set $T = \{x^t_1, \cdots, x^t_{N_t}\}$, where $x^s, x^t$ represent the source and target input sample, and $y^s \in \mathbb{R}^M$ denotes the source ground truth label with $M$ classes. $N_s$ and $N_t$ are the numbers of source samples and target samples respectively. The source domain and target domain shares the same label space $\{0, 1, \cdots, M - 1\}$ but with different distributions. The goal of UDA is to mitigate the distribution discrepancy and make accurate predictions $\{\hat{y}^t\}$ on the target domain $T$.

Overview. Our method IndUDA consists of two parts, which are illustrated in Fig 2. One is an independent feature decomposition module, while the other is an instance feature alignment module. For the Independent Feature Decomposition (IFD) module as shown in Fig.2(a), the source feature $f_s$ and target feature $f_t$, which has high-confidence pseudo-label, are first mapped to a latent space $z$ by an invertible neural network $G_{flow}$ as $z_s = G_{flow}(f_s)$, $z_t = G_{flow}(f_t)$. Then, $z_s$ and $z_t$ are split into domain-invariant parts $z^d_{s}, z^d_{t}$ and domain-specific parts $z^s_{di}, z^t_{di}$ respectively by a channel mask block. To guarantee the independence and seek semantic domain-invariant information, an instance cross-domain swap strategy is employed. If the extracted domain-invariant parts are independent of the corresponding domain-specific parts, then after swapping the domain-invariant parts, the inverted features should maintain the same class memberships as the original features. In the end, a constraint loss $L_{con}$ is used to align the domain-invariant parts between source and target domains which further reduces the correlations between the domain-invariant part and domain-specific part.

For the instance feature alignment module as shown in Fig.2(b), domain-specific parts can be considered as noise because they are not related to the classification task. Our idea is to reconstruct the ideal feature which contain no domain-specific noise to guide the training of the feature extractor. So we replace $z^s_{di}, z^t_{di}$ with a Gaussian noise $r$. Here, the reason we use a Gaussian noise $r$ instead of zero is that it can help to improve the generalization ability. Then the extracted features $f_s, f_t$ are required to be aligned to the corresponding ideal features respectively by the loss $L_{align}$, i.e., a better domain-invariant space is obtained.

3.2 Independent Feature Decomposition

To achieve independent feature decomposition, this module consists of three parts: invertible feature mapping, channel mask block and instance cross-domain swap strategy.

Invertible feature mapping. After feature extractor $F$, domain-invariant features and domain-specific features are highly correlated in feature channel dimension. So it is better to map the extracted feature $f$ into a latent space $z$ by an invertible neural network $G_{flow}$ as $z_s = G_{flow}(f_s)$, $z_t = G_{flow}(f_t)$.

Channel mask block. To split the mapped features into domain-invariant part and domain-specific part in a reason-
able way, without any doubts, the domain-invariant part is relevant to classification task [Huang et al., 2020] while domain-specific part has nothing to do with the classification performance. For simplicity, we decompose features according to feature channel dimension. The gradient of classifier respect to the $i$-th average pooled channel feature is used as an evaluation measure, which is defined as

$$y_x = \frac{\partial y_x}{\partial f_x^i}, \quad (1)$$

where $y_x$ denotes the classifier output of sample $x$ and $f_x^i$ is the $i$-th average pooled channel feature. Since the extracted feature $f_x \in \mathbb{R}^{D \times H \times W}$, where $D$ denotes the channel number, after average pooling for each channel, the gradient $y_x \in \mathbb{R}^{D \times 1 \times 1}$. The gradient distributions on channels for different samples of the same category are different. So the union of all samples is considered and domain-invariant channels are chosen for each category.

A memory bank $M_{\text{bank}} \in \mathbb{R}^{C \times D}$ is established as a channel mask, where $C$ is the category number, and all components in $M_{\text{bank}}$ are initialized with 1. For a sample $x$, we have a mask for the category $c$ and the $i$-th channel is defined as follows,

$$\text{mask}_{x}^{c,i} = \begin{cases} 1, & \text{if } y_x^i > \tau, \\ 0, & \text{else} \end{cases}, \quad (2)$$

where $\tau$ is a threshold. Then $\text{mask}_{x}^{c,i}$ is used to update $M_{\text{bank}}$. The mask of class $c$ in $M_{\text{bank}}$ is updated as

$$M_{\text{bank}}[c,i] \leftarrow M_{\text{bank}}[c,i] \odot \text{mask}_{x}^{c,i},\quad (3)$$

where $\odot$ represents the element-wise multiplication. The element of $M_{\text{bank}}$ is valued at 1 only in the case that the mask of every source sample in this class is 1, otherwise is 0 for other cases. The value of threshold $\tau$ will affect feature decomposition. If $\tau$ is too small, most of the channels are reserved and domain-specific features cannot be de-noised well; while if $\tau$ is too large, most of the domain-invariant channels will be dismissed. The value of $\tau$ is decided such that just $D/2$ channels are selected as domain invariant part. After passing all source samples, we can obtain a memory bank $M_{\text{bank}}$ for each category. For selecting semantic domain-invariant channels, we only need to query $M_{\text{bank}}$ without calculating the gradient again.

Remark. There is another way to split the feature, i.e., directly bisecting the feature and forcing one half as domain-invariant channels and the other half becomes domain-specific channels. But using this method is not easy to obtain a stable model because domain invariant channels are different for each category. It should be noted that the mask bank is only calculated based on the source samples because they have accurate labels. Experiments in Appendix also show that the masks are the same with the confident pseudo-labelled target samples.

Instance cross-domain swap strategy. First, the target samples are pseudo-labelled by a spherical K-means method as used in CAN [Kang et al., 2019]. Suppose the $c$-th cluster center in the target domain is represented as $O^c$ and only a subset is labelled as confidence samples $\tilde{T} = \{(x_t^i, y_t^i) | \text{dist}(f_t, O_t^i) < D_0\}$, where $y_t^i$ is assigned by the nearest cluster center and $D_0$ is a constant which is set to 0.05. The distance is measured as cosine dissimilarity. Besides, if the number of samples satisfying the condition in one category is less than three, this category will not be sampled in the current epoch.

After that, by invertible feature mapping, the extracted source and confidence target features $f_s$ and $f_t$ are mapped as $z_s$ and $z_t$. Then, the features $z_s$ and $z_t$ are further split into $[z_s^{di}, z_s^{ds}]$ and $[z_t^{di}, z_t^{ds}]$ by $M_{\text{bank}}$, where $z_s^{di}, z_t^{di}$ composed of channels with mask value 1. Since we assume that the domain-invariant features should follow the same distribution if we exchange $z_s^{di}$ and $z_t^{di}$ to get two features $z_s'$ and $z_t'$, their distributions should also be the same. This process can be written as follows:

$$z_s' = z_t^{di} \oplus z_s^{ds}, \quad z_t' = z_t^{di} \oplus z_s^{ds},\quad (4)$$

where $\oplus$ denotes the concatenate operation. To our analysis, the inverted features $f_s'$, $f_t'$ of $z_s'$, $z_t'$ should have the same distributions as $f_s$ and $f_t$ respectively. So for training invertible feature mapping, we propose the following swap loss

$$\min_{G_{\text{flow}}} \mathcal{L}_{\text{swap}} = D_H(f_s, f_s') + D_H(f_t, f_t'),\quad (5)$$

where $D_H$ is a function that measures the distribution difference. Here, contrastive domain discrepancy (CDD) loss [Kang et al., 2019] is used.

To further guarantee that domain-invariant parts from the source and target samples obey the same distribution, we add a constraint term $\mathcal{L}_{\text{con}}$ as

$$\min_{G_{\text{flow}}} \mathcal{L}_{\text{con}} = D_H(z_s^{di}, z_t^{di}).\quad (6)$$

We use Eq.(6) to constrain $z_s^{di}$ and $z_t^{di}$ become more compact in latent space $z$, because the CDD loss can make the intra-class distance smaller and the inter-class distance larger.

Remark. By invertible property, swap strategy reduces the correlation between domain-invariant part and domain-specific part, thus the independence of domain-invariant part is achieved in some sense. Furthermore, the distance is measured by CDD loss in the class level instead of $L_2$ loss. There are two reasons. First, the domain-invariant part for each category cannot be exactly the same, otherwise, the diversity of samples is lost. Second, CDD loss can obtain a more compact latent space.

3.3 Instance Feature Alignment

In the above section, domain-invariant part is obtained in latent space. If we project back this part to the original feature space and ask for the extracted features close to the corresponding reconstructed features, then the feature extractor can extract domain-invariant features in the original feature space. However, it is not suitable to directly drop the domain-specific channels and set them to zero. Because normalizing flow is a manifold mapping, i.e., distribution-to-distribution. If the domain-specific part is simply set as zero, the correspondence becomes one-on-one which reduces the generalization ability.
So a random variable \( r \sim \mathcal{N}(0, I) \) is employed to replace the domain-specific part, which can improve the generalization ability of the reconstructed features while also maintaining the distribution consistency between \( z_{st}^d \) and \( z_{st}^d \). In this way, we can make the domain-invariant part follow a distribution and it is also domain-agnostic because the random variable does not contain any domain-specific noise.

Formally, two denoised features with respect to two domain-invariant parts are

\[
f^s_{ide} = G^1_{flow}(z_{st}^d \oplus r), f^t_{ide} = G^1_{flow}(z_{st}^d \oplus r).
\]

An instance feature alignment method is performed to align the source and target features respectively. The alignment loss can be written as follows,

\[
\min_{F} \mathcal{L}_{align} = D_H(f_s, f^s_{ide}) + D_H(f_t, f^t_{ide}),
\]

where \( D_H \) is also the CDD loss [Kang et al., 2019]. This alignment paradigm is theoretically reasonable. For minimizing distribution divergence \( D_H(f_s, f_t) \) between the source and target domains, based on triangle inequality, we can write \( D_H(f_s, f_t) \) as follows,

\[
D_H(f_s, f_t) \leq D_H(f_s, f^s_{ide}) + D_H(f_t, f^t_{ide}) + D_H(f^s_{ide}, f^t_{ide})
\]

where the term \( D_H(f_s, f^s_{ide}) + D_H(f_t, f^t_{ide}) \) corresponds to the loss \( \mathcal{L}_{con} \); while \( D_H(f^s_{ide}, f^t_{ide}) \) corresponds to the loss \( \mathcal{L}_{align} \) in Eq.(7). Thus, \( D_H(f^s_{ide}, f^t_{ide}) \approx D_H(z_{st}^d, z_{st}^d) \) and \( G_{flow} \) is a bijection. So we only need to align the features with their corresponding reconstructed features as shown in Eq.(8). In this way, we can minimize \( D_H(f_s, f_t) \) through \( \mathcal{L}_{align} \) and \( D_H(f^s_{ide}, f^t_{ide}) \).

\[\textbf{Remark.} \] The previous methods always align the target features with the source features directly without taking independence into consideration. The performance is hugely degraded by domain-specific noise. As IndUDA aligns the source features and target features with their denoised features at the instance level, the feature extractor can efficiently obtain domain-agnostic features. Here, CDD loss is used instead of \( L_2 \) because CDD can better guarantee the consistency of class distribution.

### 3.4 Overall Loss Function

In the independent feature decomposition part, the parameters of \( G_{flow} \) are updated by gradient descent via \( L_{flow} \) as follows,

\[
\min_{F} L_{flow} = L_{swap} + L_{con}.
\]

In the instance feature alignment part, the feature extractor \( F \) is updated by minimizing \( \mathcal{L}_{align} \) in Eq. (6).

Besides, we adapt the cross entropy loss \( L_{ce} \) to maintain the performance on source model,

\[
\min_{F,C} L_{ce} = E_{x_s^i \in D_s} L_{ce}(f^s_{ide}, y_{s}^i).
\]

So the overall object function \( L_{total} \) can be written as

\[
\min_{F,G,C} L_{total} = L_{ce} + \gamma L_{flow} + \lambda L_{align},
\]

where \( \lambda \) and \( \gamma \) are trade-off parameters to weight the loss functions.

Because the feature extractor \( F \) and the invertible flow \( G_{flow} \) are two separate networks, it is not easy to optimize them together. So the training consists of two phases: first, the \( G_{flow} \) is trained, and then the trained \( G_{flow} \) is used to guide the training of \( F \). For training \( G_{flow} \), pseudo-labels are needed for target samples. The updated model will be iteratively used. In the test procedure, only the adapted networks \( F \) and \( C \) are used.

### 4 Experiments

#### 4.1 Experimental Setup

**Datasets.** We conduct experiments on three standard domain adaptation datasets. **Office-31** [Saenko et al., 2010] is a popular dataset with a total of 4652 photos, including 31 object categories in three domains: Amazon (A), DSLR (D) and Webcam (W). **Office-Home** [Venkateswara et al., 2017] is a more challenging dataset, which contains 15500 images with an average of 70 images per class. It consists of 65 categories in 4 domains: Art (A), Clipart (C), Product (P) and Real-World (R). **VisDA-2017** [Peng et al., 2018] is a simulation-to-real dataset across 12 categories. The source domain-train...
All experiments are carried out on the Pytorch platform. The invertible flow INN consists of 5 blocks, whose structure is shown in Appendix, and the subnet $\phi_{1,2,3}(\cdot)$ of each INN block consists of 3 convolution layers with LeakyReLU function only in the first layer. In order to compare the state-of-the-art methods fairly across various datasets, we use the commonly pre-trained ResNet-50/50/101 [He et al., 2016a; He et al., 2016b] on ImageNet [Deng et al., 2009] as feature extractors for Office-31, Office-Home and VisDA-2017 respectively. The classifier consists of a fully connected layer. With the exception of the batch normalization layers’ domain-specific characteristics, all network parameters are shared by the source domain and target domain data. The initial learning rate $\eta_0$ is set as 0.001 for the first convolutional layers and 0.01 for the rest including the entire INN network and no pre-trained layers in the feature extractor. We adopt the same learning rate scheduler $\eta_p = \eta_0(1 + \alpha p)^{-b}$. For Office-31 and Office-home datasets, we set $a = 10$ and $b = 0.75$, while for Visda-2017 we set $a = 10$ and $b = 2.25$. We select half of the channels as domain-invariant feature channels in channel mask block and choose 0.5, 0.3 as the value of $\gamma$ and $\lambda$ respectively.

### Competitors

We compare our method with three types of unsupervised domain adaptation methods. The first type of method adapts implicit alignment based on adversarial learning. They are CDAN [Long et al., 2018], DALN [Chen et al., 2022], CDAL [Zhou et al., 2022b], DDL [Xiao and Zhang, 2021], CLS [Liu et al., 2021a], SRADA [Wang and Zhang, 2020] and SDAT+ELS [Zhang et al., 2023]. The second type of method is explicitly alignment based on statistic moment matching. They are SUDA [Zhang et al., 2022], CaCo [Huang et al., 2022], ATDOC [Liang et al., 2021], SIDA [Zhao et al., 2022], ALDA [Chen et al., 2020] and CAN [Kang et al., 2019]. The third type of method is based on feature decomposition, which decomposes features into domain-invariant features and domain-specific features. They are DSR [Cai et al., 2019], IC²FA [Deng et al., 2021] and DTA [Lee et al., 2019].

### 4.2 Comparison with State-of-the-Art

Tables 1-2 show experiment results of our IndUDA and other state-of-the-art methods on datasets Office-home, Visda-2017, respectively. The total results on Office-31 are not shown here because the performance is saturated and limited page size, which can be found in Appendix. Especially, on Office-31, our method IndUDA reaches a mean accuracy of 91.4%, which is 0.8% higher than the best competitor (CAN). As for the Office-home, IndUDA boosts the mean accuracy to 74.3% than other state-of-the-art methods. It is 2.1% higher than the second highest method ATDOC [Liang et al., 2021]. Specifically, our results achieve the best results in 11 of the 12 tasks and only 0.1% less than the best method on task $Rw \rightarrow Pr$. On Visda-2017, IndUDA also achieves the best results compared with other state-of-the-art methods.

More specifically, for adversarial learning methods, IndUDA outperforms the best approach CDAL [Zhou et al., 2022b] by 2.4% in an average on Office-Home and visda-17 datasets. Compared with the best statistic moment matching based methods CAN [Kang et al., 2019], the result of IndUDA is 1.7% higher in an average on these two datasets. Both of these alignment methods do not consider the effectiveness of domain-specific noise, while our IndUDA makes the domain-invariant part and domain-specific part independent and denoise the domain-specific feature before alignment, which can achieve better alignment. For the feature decomposition-based domain adaptation methods, our method also outperforms the corresponding competitors, because they cannot decompose the features without domain-specific noise. All in all, the experiment results confirm our previous analysis: splitting out the domain-specific part by independence is advantageous; instance cross-domain swap strategy can obtain domain-invariant part and make the feature space more compact. These properties boost the adaptation performance.

Due to space limitations, the visual analysis is shown in Appendix. There are two basic observations. Compared with

<table>
<thead>
<tr>
<th>Method</th>
<th>Venue</th>
<th>plane</th>
<th>bcycl</th>
<th>bus</th>
<th>car</th>
<th>horse</th>
<th>knife</th>
<th>mcycl</th>
<th>person</th>
<th>plant</th>
<th>sktbrd</th>
<th>train</th>
<th>truck</th>
<th>Avg.</th>
</tr>
</thead>
<tbody>
<tr>
<td>ResNet-101</td>
<td>CVPR16</td>
<td>55.1</td>
<td>53.3</td>
<td>61.9</td>
<td>59.1</td>
<td>80.6</td>
<td>17.9</td>
<td>79.7</td>
<td>31.2</td>
<td>81.0</td>
<td>26.5</td>
<td>73.5</td>
<td>8.5</td>
<td>52.4</td>
</tr>
<tr>
<td>CAN</td>
<td>CVPR19</td>
<td>97.0</td>
<td>87.2</td>
<td>82.5</td>
<td>74.3</td>
<td>97.8</td>
<td>96.2</td>
<td>90.8</td>
<td>98.7</td>
<td>80.7</td>
<td>96.6</td>
<td>96.3</td>
<td>87.5</td>
<td>59.9</td>
</tr>
<tr>
<td>ALDA</td>
<td>AAAI20</td>
<td>93.8</td>
<td>74.1</td>
<td>82.4</td>
<td>69.4</td>
<td>90.6</td>
<td>87.2</td>
<td>89.0</td>
<td>67.6</td>
<td>93.4</td>
<td>76.1</td>
<td>87.7</td>
<td>22.2</td>
<td>77.8</td>
</tr>
<tr>
<td>ATDOC</td>
<td>CVPR21</td>
<td>93.7</td>
<td>83.0</td>
<td>76.9</td>
<td>58.7</td>
<td>89.7</td>
<td>95.1</td>
<td>84.4</td>
<td>71.4</td>
<td>89.4</td>
<td>80.0</td>
<td>86.7</td>
<td>55.1</td>
<td>80.3</td>
</tr>
<tr>
<td>SIDA</td>
<td>IJCAI22</td>
<td>95.4</td>
<td>83.1</td>
<td>77.1</td>
<td>64.6</td>
<td>94.5</td>
<td>97.2</td>
<td>88.7</td>
<td>78.4</td>
<td>93.8</td>
<td>89.9</td>
<td>85.2</td>
<td>59.4</td>
<td>84.0</td>
</tr>
<tr>
<td>SUDA</td>
<td>CVPR22</td>
<td>88.3</td>
<td>79.3</td>
<td>66.2</td>
<td>64.7</td>
<td>87.4</td>
<td>80.1</td>
<td>85.9</td>
<td>78.3</td>
<td>86.3</td>
<td>78.5</td>
<td>78.8</td>
<td>74.5</td>
<td>79.8</td>
</tr>
<tr>
<td>CaCo</td>
<td>CVPR22</td>
<td>90.4</td>
<td>80.7</td>
<td>78.8</td>
<td>57.0</td>
<td>89.8</td>
<td>87.0</td>
<td>81.3</td>
<td>79.4</td>
<td>88.7</td>
<td>81.1</td>
<td>86.8</td>
<td>63.9</td>
<td>80.9</td>
</tr>
<tr>
<td>CDAN</td>
<td>NIPS18</td>
<td>85.2</td>
<td>66.9</td>
<td>83.0</td>
<td>50.8</td>
<td>84.2</td>
<td>74.9</td>
<td>88.1</td>
<td>74.5</td>
<td>83.4</td>
<td>76.0</td>
<td>81.9</td>
<td>38.0</td>
<td>73.9</td>
</tr>
<tr>
<td>DWL</td>
<td>CVPR21</td>
<td>90.7</td>
<td>80.2</td>
<td>86.1</td>
<td>67.6</td>
<td>92.4</td>
<td>81.5</td>
<td>86.8</td>
<td>78.0</td>
<td>90.6</td>
<td>57.1</td>
<td>85.6</td>
<td>28.7</td>
<td>77.1</td>
</tr>
<tr>
<td>CLS</td>
<td>ICCV21</td>
<td>92.6</td>
<td>84.5</td>
<td>73.7</td>
<td>72.7</td>
<td>88.5</td>
<td>83.3</td>
<td>89.1</td>
<td>77.6</td>
<td>89.5</td>
<td>89.2</td>
<td>85.8</td>
<td>72.7</td>
<td>81.6</td>
</tr>
<tr>
<td>CDAL</td>
<td>ACCMM22</td>
<td>97.5</td>
<td>84.9</td>
<td>81.0</td>
<td>70.5</td>
<td>97.1</td>
<td>97.3</td>
<td>90.6</td>
<td>80.9</td>
<td>96.2</td>
<td>94.9</td>
<td>88.2</td>
<td>48.7</td>
<td>85.7</td>
</tr>
<tr>
<td>SRADA</td>
<td>IJCAI20</td>
<td>95.4</td>
<td>83.1</td>
<td>77.1</td>
<td>64.6</td>
<td>94.5</td>
<td>97.2</td>
<td>88.7</td>
<td>78.4</td>
<td>93.8</td>
<td>89.9</td>
<td>85.2</td>
<td>59.4</td>
<td>84.0</td>
</tr>
<tr>
<td>IC²FA</td>
<td>ACCMM22</td>
<td>89.7</td>
<td>70.6</td>
<td>79.8</td>
<td>84.3</td>
<td>96.5</td>
<td>72.1</td>
<td>90.4</td>
<td>65.3</td>
<td>92.7</td>
<td>63.3</td>
<td>86.5</td>
<td>36.0</td>
<td>77.3</td>
</tr>
<tr>
<td>DTA</td>
<td>ICCV19</td>
<td>93.7</td>
<td>82.2</td>
<td>85.6</td>
<td>83.8</td>
<td>93.0</td>
<td>81.0</td>
<td>90.7</td>
<td>82.1</td>
<td>95.1</td>
<td>78.1</td>
<td>86.4</td>
<td>32.1</td>
<td>81.5</td>
</tr>
</tbody>
</table>

Table 2: Comparison with state-of-the-art methods on Visda-17 dataset. Metric: per-class classification accuracy (%); Backbone: ResNet-101.
### 4.3 Further Analysis

**Ablation study on loss function.** The ablation analysis is performed on Office-31 and Office-home dataset. For Office-31, the experiments are evaluated on the tasks $A\rightarrow W$, $A\rightarrow D$, $D\rightarrow A$ and $W\rightarrow A$, as the performance on other two tasks is saturated. To explore the effectiveness of different loss functions (components), we do experiments without $L_{\text{con}}$ and without $L_{\text{swap}}$ respectively. Without $L_{\text{con}}$, it means that the domain-invariant part is obtained only via swap strategy, i.e. using the loss $L_{\text{swap}}$. While for the experiment without $L_{\text{swap}}$, it means we only align the domain-invariant part by the consistency loss $L_{\text{con}}$. It can be seen from Tab.3 that swap strategy $L_{\text{swap}}$ can reach a better alignment performance on the domain-invariant feature. The performance is 1.6% higher on each task than those without $L_{\text{swap}}$. Because the input and output of invertible feature mapping are strictly one-to-one corresponding, it adds a strong constraint such that the domain-invariant part and domain-specific part are well separated. Furthermore, from the last two rows in Tab.3, without the constrain loss $L_{\text{con}}$, the performance decreases by 1%, because the loss $L_{\text{con}}$ makes the features in the latent space more compact such that the reconstructed feature is denoised better.

**Ablation analysis on invertible neural networks.** To verify the effectiveness of using INN, we experiment with different variants, i.e., two INNs, Doublemap and One INN (IndUDA). Two INNs mean two invertible neural networks are trained corresponding to the source and target features respectively. Without the swap strategy, the features reconstructed by double mapping have more errors compared with one INN, resulting in lower accuracy.

**Sensitivity analysis of the parameters.** There are three hyper-parameters, which are $\gamma$, $\lambda$ in Eq.(11) and $\tau$ in Eq.(2), respectively. Experiments are performed on Office-31. First, with different values of $\gamma$, $\lambda$ in the range of [0.1, 2.0], as shown in Fig.3, the performance of our method consistently outperforms the baseline over a wide range and is stable as the value gets larger. Besides, the bell-shaped curves also prove the robustness of our method. According to the performance curves, we set $\gamma$ and $\lambda$ equal to 0.5, and 0.3 respectively.

Regarding $\tau$, its value is set such that $D/8$, $D/4$, $D/2$, $3D/4$, $D$ channels are selected as domain-invariant channels, where $D=2048$ is the amount of feature channels. It can be observed that the accuracy is highest when $D/2$ channels are exactly selected. The performance of our method is not good when only $D/8$ and $D/4$ channels are chosen. This case misses many domain-invariant channels and dismisses some indispensable channels for classification; while choosing $3D/4$ and $D$ channels means almost all channels are reserved and domain-specific channels are not dismissed. So half channels for the domain-invariant part and another half for the domain-specific part is a reasonable choice.

## 5 Conclusion

In this paper, we proposed a novel framework, named Independent feature decomposition and alignment for Unsupervised Domain Adaptation (IndUDA). By considering domain-specific information contained in the features as noise, and sufficiently using the property of invertible flow, a swap strategy is proposed to guarantee the separated domain-invariant and domain-specific parts in the latent space are independent. After discarding domain-specific parts, an independent alignment module is used to align the inverted ideal domain-invariant features with the original features. Thus domain-invariant features in the original feature space are obtained. Experiments on real-world benchmarks demonstrate the superiority and effectiveness of our method compared with the state-of-the-art baselines.
**Ethical Statement**

There are no ethical issues.

**References**


