Supplement: Gotta Adapt ’Em All: Joint Pixel and Feature-Level Domain Adaptation for Recognition in the Wild

Luan Tran[1] Kihyuk Sohn[2] Xiang Yu[2] Xiaoming Liu[1] Manmohan Chandraker[2,3]
[1]Michigan State University [2]NEC Labs America [3]UC San Diego

S1. A Gradient Analysis of Classification-Aware DANN

Here we provide a detailed derivation of the gradient analysis of DANN and DANN-CA presented in Section 3.2.

S1.1. Gradient for DANN

Let $\phi_d : \mathbb{R}^K \to \mathbb{R}$ be a function that generates the exponent of discriminator distribution, i.e.,

$$D(f) = \sigma\left(\phi_d(f)\right) \tag{S1}$$

where $\sigma(\cdot)$ is a logistic (sigmoid) function. Then, we get the following gradient:

$$\begin{aligned}\frac{\partial \log(1-D(f))}{\partial f} &= \frac{-1}{1-D(f)} D(f)(1-D(f)) \frac{\partial \phi_d(f)}{\partial f} && \text{(S2)}\\ &= -D(f)\frac{\partial \phi_d(f)}{\partial f} && \text{(S3)}\\ &= -D(f) w_d && \text{(S4)}\end{aligned}$$

When the discriminator is linear, i.e., $\phi_d(f) = w_d^\top f$, we end up Eq (S4), which is equivalent to the first one in Eq (7).

S1.2. Gradient for DANN-CA

Let $\phi_y : \mathbb{R}^K \to \mathbb{R}, y = 1, ..., \mathcal{Y}+1$ be a function that generates the exponent of classification distribution of DANN-CA:

$$\overline{C}(y) = \frac{\exp(\phi_y(f))}{\sum_{y'=1}^{\mathcal{Y}+1} \exp(\phi_{y'}(f))} \tag{S5}$$

The gradient of adversarial loss in Eq (5) with respect to f is written as follows:

$$\frac{\partial \log(1-\overline{C}(N+1))}{\partial f} = \frac{-1}{1-\overline{C}(N+1)} \frac{\partial \overline{C}(N+1)}{\partial f} \tag{S6}$$

and the second term of RHS is written as

$$\frac{\partial \overline{C}(N+1)}{\partial f} = \frac{\phi'_{N+1} \exp(\phi_{N+1})}{\sum_{y'=1}^{\mathcal{Y}+1} \exp(\phi_{y'})} - \frac{\exp(\phi_{N+1}) \sum_{y'=1}^{\mathcal{Y}+1} \phi'_{y'} \exp(\phi_{y'})}{\{\sum_{y'=1}^{\mathcal{Y}+1} \exp(\phi_{y'})\}^2} \tag{S7}$$

$$= \phi'_{N+1}\overline{C}(N+1) - \overline{C}(N+1) \sum_{y=1}^{\mathcal{Y}+1} \phi'_y \overline{C}(y) \tag{S8}$$

$$= \phi'_{N+1}\overline{C}(N+1)(1-\overline{C}(N+1)) - \overline{C}(N+1) \sum_{y=1}^{\mathcal{Y}} \phi'_y \overline{C}(y) \tag{S9}$$

where $\phi'_y = \frac{\partial \phi_y(f)}{\partial f}$. Plugging Eq (S9) into Eq (S6) results in the following:

$$\frac{\partial \log(1-\overline{C}(N+1))}{\partial f} = -\phi'_{N+1}\overline{C}(N+1) + \overline{C}(N+1)\sum_{y=1}^{\mathcal{Y}} \phi'_y \overline{C}(y|\mathcal{Y}) \tag{S10}$$

$$= -w_{N+1}\overline{C}(N+1) + \overline{C}(N+1)\sum_{y=1}^{\mathcal{Y}} w_y \overline{C}(y|\mathcal{Y}) \tag{S11}$$

where we assume linear classifier and discriminator, $\phi_y(f) = w_y^\top f, y = 1, ..., N+1$ to derive Eq (S11) from Eq (S10).

S2. Relation to Maximum Classifier Discrepancy

Here we provide a detailed derivation of relation between our DANN-CA and recently proposed Maximum Classifier Discrepancy (MCD) learning [22], one of the consistency-based learning frameworks [13, 26, 4], presented in Section 3.2.

S2.1. Maximum Classifier Discrepancy for Unsupervised Domain Adaptation

We briefly review the MCD learning framework for unsupervised domain adaptation. Similarly to the setting of the DANN, the MCD learning divides the classifier parameterized by deep neural networks into feature extractor ($f : \mathcal{X} \to \mathbb{R}^K$) and classifiers built on top of feature extractor. Differently, it contains two (or more) classifiers $F_i : \mathbb{R}^K \to (0,1)^{\mathcal{Y}}$ with no domain discriminator.

The learning proceeds as follows: First, two classifiers are trained (while fixing the feature extractor) to minimize the classification loss on the source domain while making maximally different prediction between classifiers on the target domain. Second, feature extractor is trained (while fixing classifiers) to minimize the classification loss on the source domain while making consistent prediction between classifiers on the target domain. The learning objective is written as follows:

$$\max_{F_1,F_2} \mathbb{E}_{(x,y)\in\mathcal{X}_S\times\mathcal{Y}}\big[\log F_1(f,y) + \log F_2(f,y)\big] + \mathbb{E}_{x\in\mathcal{X}_T} d\left(F_1(f,\cdot), F_2(f,\cdot)\right) \tag{S12}$$

$$\max_{f} \mathbb{E}_{(x,y)\in\mathcal{X}_S\times\mathcal{Y}}\big[\log F_1(f,y) + \log F_2(f,y)\big] - \mathbb{E}_{x\in\mathcal{X}_T} d\left(F_1(f,\cdot), F_2(f,\cdot)\right) \tag{S13}$$

The choice of discrepancy metric d could be diverse, and in [22] $L1$-distance $d(p_1, p_2) = \frac{1}{N}\sum_{y=1}^{N} |p_1(y) - p_2(y)|$ is used.

S2.2. Relation between DANN-CA and Maximum Classifier Discrepancy [22].

Now we derive the relation between DANN-CA and MCD learning presented in Section 3.2 with more details. Following [22], we define the two classification distributions:

$$p_1(y|x_t) = \overline{C}(y|\mathcal{Y}),\ p_2(y|x_t) = \overline{C}(y),\ y \le N+1 \tag{S14}$$

Note that two classifiers F_1 and F_2 are both represented as $(N+1)$-way classifier parameterization in DANN-CA. Using KL divergence as discrepancy metric between two distributions, we obtain following discrepancy loss:

$$-\mathrm{KL}(p_1\|p_2) = -\sum_{y}^{N+1} p_1(y)\log\frac{p_1(y)}{p_2(y)} \tag{S15}$$

$$= -\sum_{y}^{N} p_1(y)\log\frac{p_1(y)}{p_2(y)} \tag{S16}$$

$$= -\sum_{y}^{N} p_1(y)\log\frac{1}{1-\overline{C}(N+1)} \tag{S17}$$

$$= \log(1-\overline{C}(N+1)) \tag{S18}$$

where Eq (S16) is due to $\overline{C}(N+1|\mathcal{Y}) = 0$ and Eq (S17) is due to $\overline{C}(y|\mathcal{Y}) = \frac{\overline{C}(y)}{1-\overline{C}(N+1)}$ for all $y \neq N$. In other words, besides the specific choice of two classifiers and discrepancy kernels ($L1$-distance versus KL divergence), two frameworks are indeed equivalent and thus are expected to have similar empirical performances as well. Empirical comparison of UDA methods including our proposed DANN-CA, MCD [22], as well as other consistency-based methods [13, 26, 4] is left as a future work.

S3. Unsupervised Model Selection

Model selection is an important component of unsupervised domain adaptation research since, we seldom have labeled examples from the target domain for validation due to its nature. Therefore, unsupervised model selection, i.e., model selection without using labeled examples from the target domain, is an essential component for any UDA method to be useful in practice. In this section, we introduce a variant of reverse validation [28, 5], the only unsupervised model selection method to our knowledge, and compare its effectiveness in comparison to our supervised model selection protocol using 5 images per output classes.

Reverse validation [28, 5] is proposed to validate the performance of domain adaptation methods without using labeled examples from the target domain. The protocol is given as follows:

- Train domain adaptation model (forward classifier) from source to target;
- Train a "reverse" classifier from unlabeled target examples using pseudo labels predicted by the forward classifier;
- Evaluate the performance of "reverse" classifier on labeled source examples.

The intuition is that if the forward classifier works well on the target examples, then the reverse classifier will also do well on the source domain, where one can have many labeled examples.

As the procedure introduces a new reverse classifier, the selection of classification method seems important. It is suggested from [5] to use the same network architecture, possibly initialized from the same network parameters of forward classifier as reverse classifier. However, we find that this selection is not particularly attractive for the following reasons. Firstly, the reverse classifier, which is another deep neural network, is expensive and non-trivial to train, e.g., it may require additional hyperparameter tuning as two domains are not always symmetric. Secondly, deep networks are robust to noise and sometimes adding label noise improves the generalization performance of deep neural network [27]. These observations suggest limited correctness of the assumption of reverse validation whereby more accurate forward classifier leads to more accurate reverse classifier. For example, our experiment with office database shows that accuracies of reverse classifiers[1] on labeled source examples with DANN and DANN-CA as forward classifier on A$\rightarrow$W task are 66.21% and 58.57%, respectively, while the performance of forward classifier on target examples are 72.33% and 77.38%. Note that the performance of reverse classifier using the ground-truth labels as self-labeled target set is only 46.43%, which verifies the effectiveness of noisy labels in training deep neural network.

Instead, we propose few alternatives that are much simpler and more efficient to evaluate based on non-parameteric classifiers. We summarize our proposed unsupervised validation metrics below.

1. k-nearest neighbor: we use k-nearest neighbor classifier using learned representation of forward model f and predicted labels $C(f)$ (or $\overline{C}(f|\mathcal{Y})$ for DANN-CA) on target examples by forward model. The performance measure evaluated on labeled source data is given as follows:

$$\text{ACC}_{\text{kNN}} = \mathbb{E}_{(x,y)\in\mathcal{X}_{\text{S}}\times\mathcal{Y}_{\text{S}}}\mathbf{1}\{y = \arg\max_{\tilde{y}} \frac{1}{k}\sum_{\tilde{x}\in\text{kNN}(x)} C(f(\tilde{x}),\tilde{y})\} \tag{S19}$$

   We use $k = 5$ for all our experiments.

2. mAP: we use an average precision (AP) of labeled source examples with label-predicted target examples via forward classifier. The performance measure is given as follows:

$$\text{mAP} = \mathbb{E}_{(x,y)\in\mathcal{X}_{\text{S}}\times\mathcal{Y}_{\text{S}}} AP(x, y|\{x_t, \arg\max_{y'} C(f(x_t), y')\}_{x_t\in\mathcal{X}_{\text{T}}}) \tag{S20}$$

The results with our proposed model selection methods are found in Table S1. We observe that non-parameteric classifiers defined on learned representation can find models that are consistent with test set performance of the models chosen by supervised model selection method using 5 images per class. Although we find these unsupervised metrics effective, we also observe significant performance drop for some models selected by 5-NN or mAP (e.g., M6–M7, M12–M13). We believe that unsupervised model selection in deep domain adaptation is not yet solved and requires significant more investigation, both from empirical and theoretical perspectives, which is beyond the scope of our work and will leave them as a future work.

[1]For simplicity, we train a classifier of the same network configuration to forward classifier with self-labeled target domain examples, but without adaptation loss.

ID	Persp.	Photo.	Feature	5/cls (sup.)			5-NN (unsup.)			mAP (unsup.)		
				Top-1	Day	Night	Top-1	Day	Night	Top-1	Day	Night
M1	Baseline (web only)			54.98	72.67	19.87		–			–	
M2	Supervised (web + SV)			98.63	98.92	98.05		–			–	
M3	AF	–	–	59.73	75.78	27.87	58.88	75.27	26.35	58.89	75.64	25.64
M4	KF	–	–	61.55	77.98	28.92	60.87	76.70	29.45	60.47	76.56	28.52
M5	MKF	–	–	64.30	78.62	35.87	61.63	75.53	34.04	64.37	78.67	35.99
M6	–	CycleGAN	–	64.32	77.01	39.12	60.92	73.55	35.87	61.25	73.95	36.02
M7	–	AC-CGAN	–	67.30	78.20	45.66	67.44	78.53	45.41	64.52	76.12	41.48
M8	MKF	CycleGAN	–	71.21	81.54	50.68	69.42	79.59	49.23	70.85	81.95	48.82
M9	MKF	AC-CGAN	–	79.71	84.10	70.99	74.98	79.70	65.62	78.80	83.18	70.09
M10	–	–	DANN	60.40	75.56	30.31	58.15	73.97	26.74	60.05	75.52	29.32
M11	–	–	DANN-CA	75.83	76.73	74.05	75.01	76.53	71.99	75.40	76.51	73.19
M12	MKF	–	DANN-CA	80.40	82.50	76.22	77.26	82.44	66.98	75.85	82.42	62.82
M13	–	AC-CGAN	DANN-CA	80.24	82.15	76.44	77.69	82.17	68.78	77.91	82.15	69.50
M14	MKF	AC-CGAN	DANN-CA	**84.20**	**85.77**	**81.10**	**83.78**	**85.54**	**80.27**	**83.82**	**85.56**	**80.37**

Table S1: Car recognition accuracy on surveillance images of CompCars dataset of our recognition system with different combinations of components evaluated by supervised and unsupervised model selection methods. We consider pixel-based (AF), keypoint-based (KF) and with mask (MKF) for perspective transformation, CycleGAN and attribute-conditioned CycleGAN, and DANN, DANN-CA as variations.

S4. Implementation Details

In this section, we describe implementation details of individual components. All components are implemented in Torch [3].

S4.1. Appearance Flow Estimation Networks

AFNet has encoder-decoder structure, which is visualized in Fig. S1. AFNet takes a source image and target viewpoint as input, where an image of size 256×256 is fed to a convolutional encoder to produce a 2048-dimensional vector and it is concatenated with 512-dimensional vector generated from the latent code via viewpoint encoder. 2560-dimensional concatenated vector is fed to decoder, which is constructed with fractionally-strided convolution layers, to generate flow representation of size $256\times256\times2$. Finally, a source image is warped via appearance flow based on bilinear sampling [9, 29][2] to predict a target image. All convolution layers use 3×3 filters, meanwhile filters of fractionally-strided convolution layers have size of 4×4. AFNet is trained using Adam optimizer [11] with the learning rate of 0.0003 and batch size of 256.

KFNet architecture is inherited from AFNet and shares the decoder architecture and viewpoint encoder. To accommodate sparse keypoints as the input, the entire image encoder is replaced by the keypoint encoder, consisting of two fully connected layers with 256 and 2048 output neurons, respectively. KFNet is trained to optimize Eq (15) with $\lambda = 1$. Other hyperparameters such as the learning rate are the same as those used for AFNet training.

S4.2. Attribute-conditioned CycleGAN

The network architecture for generators and discriminators are illustrated in 5(a). The images of size 256×256 are used across input or output of generators and discriminators. UNet architecture [8] is used for both generators G and F while we feed the attribute code a in the middle of the generator network. The 70×70 patchGAN discriminator [8] is used that generates 26×26-dimensional output for real/fake discrimination. The discriminator of conditional GAN [18] is used where D takes attribute code as an additional input to the real or generated images.[3] One can consider a multi-way discriminator [25, 2] that discriminates not only between real or generated but also between different attribute configurations, but we didn't find it effective in our experiment.

We train using Adam optimizer with learning rate of 0.0002 and the batch size of 32 for all networks. In addition, we adopt two techniques from recent works to stabilize training procedure. For example, we replace the negative log likelihood

[2]https://github.com/qassemoquab/stnbhwd

[3]In our experiments, we maintain two sets of parameters for day and night attribute configurations in a similar manner to AC-CycleGAN with unshared generators in Fig. 5(b).

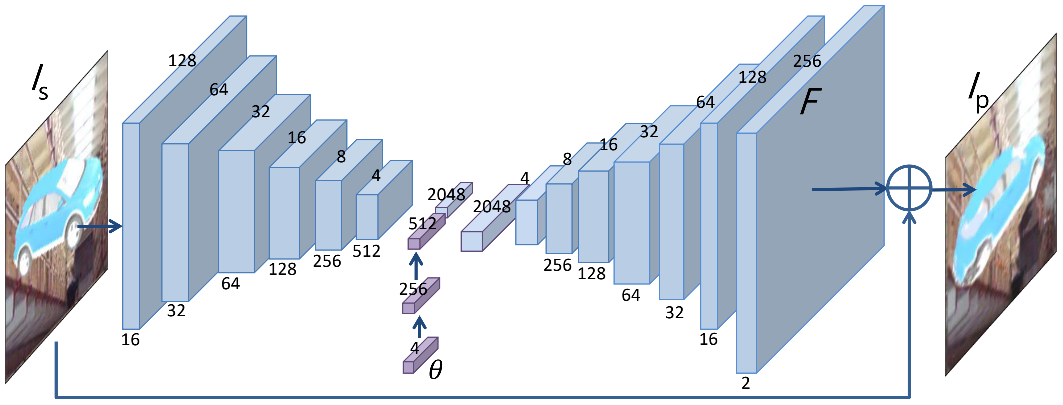


Figure S1: AFNet architecture. AFNet receives source image I_s and the target perspective θ (e.g., 4-dimensional one hot vector for elevation from $0^\circ\sim30^\circ$) as input and generates the flow field F to synthesize image I_p through bilinear sampling.

objective of discriminator by a least square loss [17, 30]. Furthermore, we adopt historical buffer strategy [23] that updates the discriminator not only using generated images with the current generator but also with the generated images from the previous updates. We maintain an image buffer that stores the 1000 previously generated images for each generator and randomly select 32 images in the buffer to update discriminator.

S4.3. Domain Adversarial Neural Networks

The ImageNet pretrained ResNet-18 [7][4] fine-tuned on the CompCars web dataset is used as our baseline network. The dimension of the last fully-connected layer is 512. The linear classifier (512 − 431) is used for all models. For discriminator, we try both linear (512 - 1) and MLP with different depth (512 - 320$\times d$ - 1, $d = 1, ..., 4$) discriminators. The validation accuracy is given in Table S2 and we decide to use 3-layer ($d = 2$) MLP discriminator. Therefore, we employ linear discriminator (512 - 432) for our proposed DANN-CA and DANN-EM, while using MLP discriminator for standard DANN. We augment the classifier of the baseline network by adding one more column to construct the weight matrices for classifiers of DANN-CA and DANN-EM. The 432nd column of the weight matrix is initialized by averaging the previous 431 weight vectors, i.e., $w_{i,432} = \frac{1}{431}\sum_{k=1}^{431} w_{i,k}, i = 1, ..., 512$.

For data preprocessing, we crop and scale web images into 256$\times$256 using provided bounding boxes while maintaining the aspect ratio. Since they are already cropped, surveillance images are simply scaled into 256$\times$256. We further crop an image of size 224$\times$224 at random location of an image of size 256$\times$256 with a random horizontal flip to feed to our feature extractor. All models are trained by updating the classifier/discriminator and CNN parameters in turn. Adam optimizer is used for training with the learning rate of 0.00001, which is equivalent to the final learning rate of the fine-tuned model on CompCars web dataset. In addition to λ in Eq (3) and (5) that balances classification loss and domain adversarial loss for updating parameters of feature extractor, we also tune learning rates of classifier and discriminator separately. Specifically, we augment Eq (2) and (4) as follows:

$$\max_{\theta_d}\{\mathcal{L}_{\text{D}} = \mathbb{E}_{\mathcal{X}_{\text{S}}} \log(1{-}D(f)) + \beta\mathbb{E}_{\mathcal{X}_{\text{T}}} \log D(f)\} \tag{S21}$$

$$\max_{\theta_c}\{\bar{\mathcal{L}}_{\text{C}} = \mathbb{E}_{\mathcal{X}_{\text{S}}} \log \bar{C}(y) + \beta\mathbb{E}_{\mathcal{X}_{\text{T}}} \log \bar{C}(N{+}1)\} \tag{S22}$$

We apply regularization coefficient β to loss induced by the target examples. When $\beta = 1$, it becomes equivalent to that of [5].[5] In these experiments, we find that $\beta = \frac{1}{N}$ is a good starting point for hyperparameter search of DANN-CA, where $N = 431$ is the number of classes and we finally fix $\beta = 0.001$ for models used in experiments on CompCars dataset. Due to small β, we increase λ for DANN-CA to backpropagate sufficient amount of gradient from adversarial loss. We also tune β for DANN from $\{100, 10, 1, 0.1, 0.01, 0.001\}$, but we don't observe significant performance difference. As a result we fix $\beta = 1$ for DANN. The optimal setting of other hyperparameters are reported in Table S3.

[4] https://github.com/facebook/fb.resnet.torch/tree/master/pretrained

[5] We set $\beta = 1$ for experiments on office database in Section S5.2 following the implementation by [5] to inherit most of the training protocol such as hyperparameter setting.

	linear	MLP with $d = 1$	MLP with $d = 2$	MLP with $d = 3$	MLP with $d = 4$
Accuracy	58.40±0.59	59.11±0.79	60.01±0.74	59.23±0.66	59.45±0.68

Table S2: Car recognition accuracy on SV validation set of CompCars of DANNs with different discriminator architectures.

DANN	DANN-CA
$\beta = 1, \lambda = 0.01$	$\beta = 0.001, \lambda = 100$

Table S3: Optimal hyperparameter settings on CompCars dataset.

S5. Details on Section 5.4: "Evaluation on UDA Benchmark"

We provide more details for the evaluation of DANNs on standard UDA benchmarks. As presented in Section 5.4, we evaluate on four tasks of digits and traffic sign recognition problems [5] and six tasks of office object recognition problems [21]. The details, such as task description or experimental results, of individual experiments are discussed below.

S5.1. Digits and Traffic Signs

S5.1.1 Task Description.

We start the section by task description and model selection. We note that supervised model selection using a subset of labeled target examples is used for this experiment inspired by [1].

1. MNIST→MNIST-M: MNIST-M is a variation of MNIST with color-transformed foreground digits over natural images in the background. Following [6], we augment source data by inverting pixel-values from 0 to 255 and vice versa, thus doubling the volume of source data. Overall, $120K(= 60K \times 2)$ labeled source images, $50K$ unlabeled target images for training, $1,000$ labeled target images for validation, $9,001$ labeled target images for testing are used.

2. Synthetic Digits→SVHN: Synthesized digits [5] are used as labeled training examples to recognize digits in street view house number dataset (SVHN) [20]. Unlike other works, we use extra unlabeled images of SVHN dataset to train adaptation models. Overall, $479,400$ labeled source images, $581,131$ unlabeled target images for training, $1,000$ labeled target images for validation, $26,032$ labeled target images for testing are used.

3. SVHN→MNIST: SVHN is used as a source and MNIST is used as a target. Overall, $73,257$ labeled source images, $50K$ unlabeled target images for training, $1,000$ labeled target images for validation, $10K$ labeled target images for testing are used.

4. MNIST→SVHN: MNIST is used as a source and SVHN is used as a target. Overall, $50K$ labeled source images, $73,257$ unlabeled target images for training, $1,000$ labeled target images for validation, $26,032$ labeled target images for testing are used.

5. Synthetic Signs→GTSRB: In this task we recognize traffic signs from german traffic sign recognition benchmark (GTSRB) [24] by adapting from labeled synthesized images [19]. In total, $90K$ labeled source images, $35K$ unlabeled target images for training, 430 labeled target images for validation, $12,569$ labeled target images for testing are used. Unlike aforementioned tasks with 10-way classification using 32×32 images as input, this task is 43-way classification and input images are of size 40×40.

For all tasks, we apply the same data preprocessing of channel-wise mean and standard deviation normalization per example [6], i.e.,

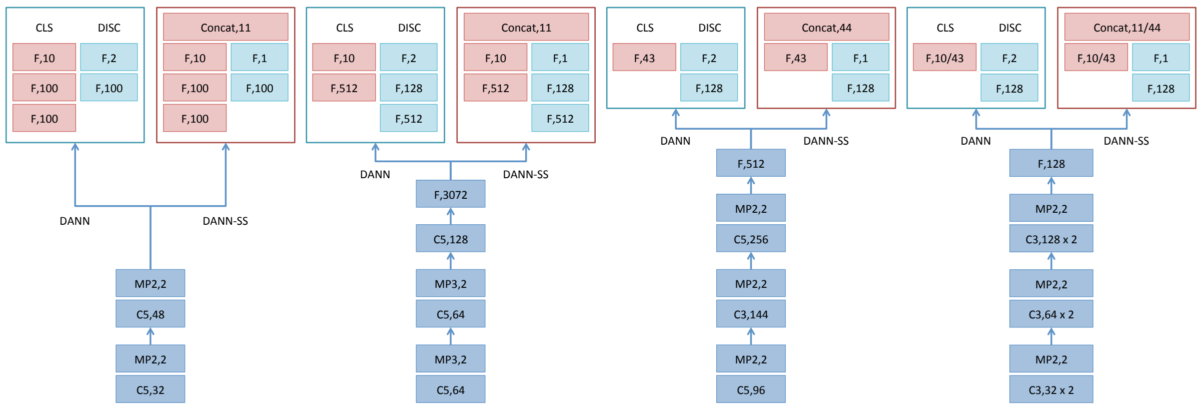
$$\tilde{x}_{i,j,c} = \frac{(x_{i,j,c} - \bar{x}_c)}{\hat{x}_c} \tag{S23}$$

where $\bar{x}_c = \frac{1}{w \times h} \sum_{i=1}^{w} \sum_{j=1}^{h} x_{i,j,c}$ and $\hat{x}_c = \sqrt{\frac{1}{(w \times h)-1} \sum_{i=1}^{w} \sum_{j=1}^{h} (x_{i,j,c} - \bar{x})^2}$.

We experiment with shallow (2∼3 convolution layers) [5] and deep (6 convolution layers) [6] network architectures as described in Fig. S2. The shallow network architectures are inspired by [5] and share the same convolution and pooling architecture, but the classifier and discriminator architectures are slightly different. Similarly, convolution and pooling architecture of deep network is the same as that of [6] but classifier and discriminator are of our own design.

Method	# val. set	network	M→MM	S→S	S→M	M→S	S→G
RevGrad [5]	0	shallow	76.67	91.09	73.85	–	88.65
DSN [1]	1000/430	shallow	83.2	91.2	82.7	–	93.1
ADA [6]	–	deep	89.53	91.86	**97.6**	–	97.66
source only			68.28$_{\pm 0.29}$	87.22$_{\pm 0.18}$	68.39$_{\pm 0.79}$	59.80$_{\pm 0.57}$	95.63$_{\pm 0.13}$
DANN	1000/430	shallow	88.62$_{\pm 0.29}$	88.07$_{\pm 0.16}$	92.34$_{\pm 0.88}$	75.48$_{\pm 2.10}$	97.33$_{\pm 0.10}$
DANN-CA			**90.41**$_{\pm 0.20}$	**93.32**$_{\pm 0.12}$	**94.15**$_{\pm 1.42}$	**82.96**$_{\pm 0.90}$	**98.47**$_{\pm 0.09}$
source only			67.90$_{\pm 0.95}$	87.05$_{\pm 0.22}$	63.74$_{\pm 0.68}$	62.44$_{\pm 0.52}$	94.53$_{\pm 0.14}$
DANN	1000/430	deep	**98.00**$_{\pm 0.07}$	92.24$_{\pm 0.13}$	88.70$_{\pm 0.33}$	82.30$_{\pm 1.15}$	97.38$_{\pm 0.13}$
DANN-CA			**98.03**$_{\pm 0.06}$	**94.47**$_{\pm 0.06}$	**96.23**$_{\pm 0.14}$	**87.48**$_{\pm 1.31}$	**98.70**$_{\pm 0.06}$

Table S4: Evaluation on digit and traffic sign adaptation tasks, such as MNIST [14] to MNIST-M [5] (M→MM), Synthetic Digits [5] to SVHN (S→S), SVHN to MNIST (S→M), MNIST to SVHN (M→S), or Synthetic Signs [19] to GTSRB [24] (S→G). Experiments are executed for 10 times with different random seeds and mean test set accuracy and standard error are reported. For each network architecture, the best performers and the ones within standard error are bold-faced. Finally, the best performers across different architectures are colored in red.



(a) M→MM (b) S→S, S→M, M→S (c) S→G (d) deep (all)

Figure S2: (a-c) Shallow [5] and (d) deep [6] network architectures for digit and traffic sign adaptation tasks. Three different shallow architectures are used for different tasks following [5]. ReLU activation is applied followed by convolutional and fully-connected layers except for the last fully-connected layer connected to classifier or discriminator.

S5.1.2 Results.

The summary results are provided in Table S4. We train 10 models with different random seeds per method and task and report the mean test set error and standard error. When there is a tie in validation performance between models with different sets of hyperparameter or at different training epochs, which happens quite frequently since we are using small number of validation examples, we report the average test set performance of the models. The proposed DANN-CA significantly improves the performance upon standard DANN on most tasks with both shallow and deep network architectures, achieving state-of-the-art results on 4 out of 5 tasks.

S5.2. Office Database

S5.2.1 Task Description.

The office database [21] is composed of three datasets, such as Amazon, Webcam, or DSLR, where each dataset contains images of 31 object categories from different sources. The number of images for each dataset is 2817, 795 and 498, respectively. Individual dataset is considered as one domain and six adaptation tasks are experimented in total. We note that the office database is not particularly suitable to demonstrate the effectiveness of our proposed joint pixel and feature-level adaptation framework since there is no obvious way to inject pixel-level insights, such as 3D shape or lighting variations. In addition, as

AlexNet								
Method	Val.	A→W	D→W	W→D	A→D	D→A	W→A	Avg
RevGrad [5]	RV	73.0	96.4	99.2	–	–	–	
RTN [16]	sup-1	73.3	96.8	99.6	71.0	50.5	51.0	73.7
CDAN-RM [15]	IWCV	77.9$\pm$0.3	96.9$\pm$0.2	100$\pm$0.0	74.6$\pm$0.2	55.1$\pm$0.3	57.5$\pm$0.4	77.0
CDAN-M [15]	IWCV	77.6$\pm$0.2	97.2$\pm$0.1	100$\pm$0.0	73.0$\pm$0.1	57.3$\pm$0.2	56.1$\pm$0.3	76.9
DANN	5-NN	72.10$\pm$0.70	96.29$\pm$0.06	99.45$\pm$0.05	70.97$\pm$0.49	51.06$\pm$0.41	50.83$\pm$0.37	73.45
	mAP	72.33$\pm$0.61	96.43$\pm$0.11	99.76$\pm$0.04	70.96$\pm$0.42	51.33$\pm$0.34	51.23$\pm$0.49	73.67
	sup-1	72.41$\pm$0.70	96.42$\pm$0.12	99.54$\pm$0.09	70.66$\pm$0.74	50.95$\pm$0.33	50.74$\pm$0.39	73.45
	oracle	73.64$\pm$0.51	96.86$\pm$0.10	99.92$\pm$0.04	72.09$\pm$0.55	51.98$\pm$0.17	51.91$\pm$0.32	74.40
DANN-CA	5-NN	77.23$\pm$1.37	96.87$\pm$0.03	99.56$\pm$0.12	74.10$\pm$0.93	59.23$\pm$0.62	57.89$\pm$0.81	77.48
	mAP	77.38$\pm$1.32	97.11$\pm$0.05	99.60$\pm$0.06	74.10$\pm$0.94	59.53$\pm$0.68	57.83$\pm$0.85	77.59
	sup-1	77.31$\pm$1.52	97.00$\pm$0.07	99.68$\pm$0.13	73.69$\pm$1.00	58.79$\pm$0.80	57.31$\pm$0.89	77.30
	oracle	78.09$\pm$1.46	97.28$\pm$0.03	99.88$\pm$0.11	74.58$\pm$0.88	59.70$\pm$0.68	58.20$\pm$0.80	77.95

Table S5: Evaluation on six adaptation tasks of Office benchmark using AlexNet. For each model and task, we report four numbers using different model selection mechanisms such as (first row) 5-NN classifier or (second row) mAP for reverse validation (RV) on source data, (third row) one labeled target example per class, or (fourth row) oracle selection via test set accuracy, which serves as an upper bound to aforementioned validation methods. All experiments are conducted 5 times with different random seeds and the mean accuracy and standard error are reported.

discussed in [1], the dataset might be limited as there exists considerable amount of high-level variations such as label noise and the number of examples for training deep adaptation networks is not sufficient.

Nevertheless, the dataset is still useful to demonstrate the effectiveness of our proposed feature-level DA methods, such as DANN-CA. We follow the training protocol of [5], where ImageNet-pretrained AlexNet [12] is used to initialize the network parameters while the last fully-connected layer (4096 - 1000) is replaced into shared bottleneck layer (4096 - 256) followed by classifier (256 - 31) and discriminator (256 - 1024 - 1024 - 1). We also performed the same experiments with ImageNet-pretrained ResNet-50 [7] following the protocol of [15]. We use relatively shallower network architecture for classifier and discriminator, where we first replace the last fully-connected layer (2048 - 1000) into shared bottleneck layer (2048 - 256) followed by classifier (256 - 31) and discriminator (256 - 256 - 1). For DANN-CA, the output of classifier and discriminator are concatenated to form a unified classifier.

We optimize networks using momentum SGD with "inv" learning rate decay policy of Caffe [10]. We evaluate on the fully transductive setting [5, 16], where all source and target examples are used for the training of deep networks.

ResNet-50								
Method	Val.	A→W	D→W	W→D	A→D	D→A	W→A	Avg
RevGrad [15]	IWCV	82.0$\pm$0.4	96.9$\pm$0.2	99.1$\pm$0.1	79.7$\pm$0.4	68.2$\pm$0.4	67.4$\pm$0.5	82.2
CDAN-RM [15]	IWCV	93.0$\pm$0.2	98.4$\pm$0.2	100$\pm$0.0	89.2$\pm$0.3	70.2$\pm$0.4	69.4$\pm$0.4	86.7
CDAN-M [15]	IWCV	93.1$\pm$0.1	98.6$\pm$0.1	100$\pm$0.0	93.4$\pm$0.2	71.0$\pm$0.3	70.3$\pm$0.3	87.7
DANN	5-NN	86.29$\pm$0.28	96.95$\pm$0.10	98.01$\pm$0.12	83.99$\pm$0.45	66.58$\pm$0.40	67.08$\pm$0.12	83.15
	mAP	86.42$\pm$0.34	96.81$\pm$0.28	97.91$\pm$0.20	84.10$\pm$0.51	67.73$\pm$0.61	67.10$\pm$0.25	83.35
	sup-1	85.97$\pm$0.51	96.87$\pm$0.19	97.94$\pm$0.14	84.12$\pm$0.50	67.63$\pm$0.73	66.78$\pm$0.33	83.22
	oracle	86.97$\pm$0.24	97.84$\pm$0.16	99.00$\pm$0.06	85.50$\pm$0.38	68.65$\pm$0.58	67.67$\pm$0.09	84.27
DANN-CA	5-NN	91.47$\pm$0.32	98.19$\pm$0.05	99.43$\pm$0.02	89.32$\pm$0.65	69.59$\pm$0.21	69.09$\pm$0.16	86.18
	mAP	91.47$\pm$0.32	98.26$\pm$0.11	99.52$\pm$0.04	89.28$\pm$0.61	70.11$\pm$0.17	69.34$\pm$0.21	86.33
	sup-1	91.35$\pm$0.36	98.24$\pm$0.07	99.48$\pm$0.10	89.94$\pm$0.41	69.63$\pm$0.40	68.76$\pm$0.40	86.23
	oracle	92.20$\pm$0.26	98.47$\pm$0.04	99.60$\pm$0.00	90.64$\pm$0.20	70.64$\pm$0.19	69.70$\pm$0.22	86.88

Table S6: Evaluation on six adaptation tasks of Office benchmark using ResNet-50. The same experimental protocol is employed to that using AlexNet. We also transfer the hyperparameters for each task from experiments usign AlexNet except that early stopping is done with respective model selection metrics.

ID	Persp.	Photo.	Feature	Top-1	Day	Night
M7	–	AC-CGAN (shared)	–	67.30	78.20	45.66
		AC-CGAN (unshared)		70.03	78.81	52.60
M9	MKF	AC-CGAN (shared)	–	79.71	84.10	70.99
		AC-CGAN (unshared)		77.75	82.76	67.79
M14	MKF	AC-CGAN (shared)	DANN-CA	84.20	85.77	81.10
		AC-CGAN (unshared)		**84.38**	**85.81**	**81.56**

Table S7: Comparison between AC-CGANs with shared and unshared parameters across generators and discriminators for car model recognition accuracy on CompCars Surveillance dataset.

S5.2.2 DANN-CA with Reverse Gradient.

To reduce an effort of additional hyperparameter search, we extend our proposed joint parameterization of classifier and discriminator for unsupervised domain adaptation to reverse gradient [5], a pioneering method of domain adversarial neural network. The loss formulation is similar to that of standard DANN in Eq (3) with a slight modification as follows:

$$\max_{\theta_f}\{\mathcal{L}_{\text{F}}{=}\mathcal{L}_{\text{C}}{-}\lambda\mathcal{L}_{\text{D}}{=}\mathcal{L}_{\text{C}}{-}\lambda\{\mathbb{E}_{\mathcal{X}_{\text{S}}}\log(1{-}D(f(x)))+\mathbb{E}_{\mathcal{X}_{\text{T}}}\log D(f(x))\}\} \tag{S24}$$

The losses for classifier and discriminator remain the same as in Eq (1) and (2). The negative sign on the adversarial loss in Eq (S24) amounts to reversing (and scaling with λ) the gradient before further backpropagating through f. This allows the entire network including classifier and discriminator as well as feature extractor to be trained end-to-end without alternating update. Besides the negative sign, we also notice that there is an additional source-to-target confusion term, $-\mathbb{E}_{\mathcal{X}_{\text{S}}}\log(1-D(f(x)))$, which we find playing an important role in this experiment. Inspired by our analysis, we use following formulations of DANN and DANN-CA in this experiment:

$$\max_{\theta_f}\{\mathcal{L}_{\text{F}}{=}\mathcal{L}_{\text{C}}+\lambda\{\mathbb{E}_{\mathcal{X}_{\text{S}}}\log D(f(x))+\mathbb{E}_{\mathcal{X}_{\text{T}}}\log(1-D(f(x)))\}\} \tag{S25}$$

$$\max_{\theta_f}\{\widetilde{\mathcal{L}}_{\text{F}}{=}\mathbb{E}_{\mathcal{X}_{\text{S}}}\log\widetilde{C}(y|\mathcal{Y})+\lambda\{\mathbb{E}_{\mathcal{X}_{\text{S}}}\log\widetilde{C}(N{+}1)+\mathbb{E}_{\mathcal{X}_{\text{T}}}\log(1{-}\widetilde{C}(N{+}1))\}\} \tag{S26}$$

Note that instead of having $-\mathcal{L}_{\text{D}}$ we define adversarial losses by flipping source and target labels. As a result, model components are still trained alternatively between classifiers and feature extractor. Nonetheless, this allows us to transfer most of the hyperparameters from RevGrad implementation [5][6] including those related to SGD such as learning rate or its scheduler. We perform few hyperparameter searches for λ starting from 0.1 as suggested by [5].

S5.2.3 Results.

The mean accuracy and standard error of the standard DANN and DANN-CA models trained with 5 different random seeds are reported in Table S5 and S6 using AlexNet and ResNet-50 as base networks, respectively. The proposed DANN-CA improves upon the standard DANN by a significant margin. Moreover, the model demonstrates comparable performance to the state-of-the-art method [15] on both experiments using AlexNet and ResNet-50 as backbone CNNs.

S6. AC-CGAN with Unshared Parameters

As we have small number of attribute configurations for lighting (e.g., day and night), it is affordable to use generator networks with unshared parameters. This is equivalent to having one generator for each lighting condition while the attribute code acts as a switch in selecting the respective output to attribute condition for inverse generator or discriminator. The network architecture is illustrated in Fig. 5(b). Note that this is equivalent to having one generator for each lighting condition and therefore each CycleGAN can be trained independently if we further assume unshared networks for discriminator and inverse generator. We conduct experiments on AC-CGAN with unshared parameters for all generators and discriminators and report the car model recognition accuracy in Table S7. We observe some improvement in recognition accuracy with unshared

[6] https://github.com/ddtm/caffe/tree/grl

models; for example, M7 or M14 with unshared parameters achieve higher accuracy than the ones with shared parameters. On the other hand, M9 with unshared parameters performs a bit lower than the one with shared parameters.

We visualize in Fig. S3 and S4 the photometric transformed images by AC-CGAN in both versions of shared and unshared parameters. Besides slight performance improvement for AC-CGAN with unshared parameters, we do not observe significant qualitative difference comparing to AC-CGAN with shared parameters. Eventually, we believe that the model with shared parameters is more promising for further investigation considering the expansibility of the methods with large number of attribute configurations as well as other interesting properties such as continuous interpolation between attribute configurations.

References

[1] Konstantinos Bousmalis, George Trigeorgis, Nathan Silberman, Dilip Krishnan, and Dumitru Erhan. Domain separation networks. In *NeurIPS*, 2016. 6, 7, 8

[2] Xi Chen, Yan Duan, Rein Houthooft, John Schulman, Ilya Sutskever, and Pieter Abbeel. Infogan: Interpretable representation learning by information maximizing generative adversarial nets. In *NeurIPS*, 2016. 4

[3] R. Collobert, K. Kavukcuoglu, and C. Farabet. Torch7: A Matlab-like environment for machine learning. In *BigLearn, NeurIPS Workshop*, 2011. 4

[4] Geoffrey French, Michal Mackiewicz, and Mark Fisher. Self-ensembling for visual domain adaptation. In *ICLR*, 2018. 2

[5] Yaroslav Ganin, Evgeniya Ustinova, Hana Ajakan, Pascal Germain, Hugo Larochelle, François Laviolette, Mario Marchand, and Victor Lempitsky. Domain-adversarial training of neural networks. *Journal of Machine Learning Research*, 17(59):1–35, 2016. 3, 5, 6, 7, 8, 9

[6] Philip Haeusser, Thomas Frerix, Alexander Mordvintsev, and Daniel Cremers. Associative domain adaptation. In *ICCV*, 2017. 6, 7

[7] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, 2016. 5, 8

[8] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A Efros. Image-to-image translation with conditional adversarial networks. In *CVPR*, 2017. 4

[9] Max Jaderberg, Karen Simonyan, Andrew Zisserman, et al. Spatial transformer networks. In *NeurIPS*, 2015. 4

[10] Yangqing Jia, Evan Shelhamer, Jeff Donahue, Sergey Karayev, Jonathan Long, Ross Girshick, Sergio Guadarrama, and Trevor Darrell. Caffe: Convolutional architecture for fast feature embedding. In *Proceedings of the 22nd ACM international conference on Multimedia*, pages 675–678. ACM, 2014. 8

[11] Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014. 4

[12] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *NeurIPS*, 2012. 8

[13] Samuli Laine and Timo Aila. Temporal ensembling for semi-supervised learning. In *ICLR*, 2017. 2

[14] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998. 7

[15] Mingsheng Long, Zhangjie Cao, Jianmin Wang, and Michael I. Jordan. Domain adaptation with randomized multilinear adversarial networks. *CoRR*, abs/1705.10667, 2017. 8, 9

[16] Mingsheng Long, Han Zhu, Jianmin Wang, and Michael I Jordan. Unsupervised domain adaptation with residual transfer networks. In *NeurIPS*, 2016. 8

[17] Xudong Mao, Qing Li, Haoran Xie, Raymond YK Lau, and Zhen Wang. Multi-class generative adversarial networks with the l2 loss function. *arXiv preprint arXiv:1611.04076*, 2016. 5

[18] Mehdi Mirza and Simon Osindero. Conditional generative adversarial nets. In *NeurIPS Workshop*, 2014. 4

[19] Boris Moiseev, Artem Konev, Alexander Chigorin, and Anton Konushin. Evaluation of traffic sign recognition methods trained on synthetically generated data. In *International Conference on Advanced Concepts for Intelligent Vision Systems*. Springer, 2013. 6, 7

[20] Yuval Netzer, Tao Wang, Adam Coates, Alessandro Bissacco, Bo Wu, and Andrew Y Ng. Reading digits in natural images with unsupervised feature learning. In *NeurIPS Workshop*, 2011. 6

[21] Kate Saenko, Brian Kulis, Mario Fritz, and Trevor Darrell. Adapting visual category models to new domains. In *ECCV*, 2010. 6, 7

[22] Kuniaki Saito, Kohei Watanabe, Yoshitaka Ushiku, and Tatsuya Harada. Maximum classifier discrepancy for unsupervised domain adaptation. In *CVPR*, June 2018. 2

[23] Ashish Shrivastava, Tomas Pfister, Oncel Tuzel, Joshua Susskind, Wenda Wang, and Russell Webb. Learning from simulated and unsupervised images through adversarial training. In *CVPR*, 2017. 5

[24] Johannes Stallkamp, Marc Schlipsing, Jan Salmen, and Christian Igel. The german traffic sign recognition benchmark: a multi-class classification competition. In *IJCNN*, 2011. 6, 7

[25] Yaniv Taigman, Adam Polyak, and Lior Wolf. Unsupervised cross-domain image generation. In *ICLR*, 2017. 4

[26] Antti Tarvainen and Harri Valpola. Mean teachers are better role models: Weight-averaged consistency targets improve semi-supervised deep learning results. In *NeurIPS*, 2017. 2

[27] Lingxi Xie, Jingdong Wang, Zhen Wei, Meng Wang, and Qi Tian. Disturblabel: Regularizing cnn on the loss layer. In *CVPR*, 2016. 3

[28] Erheng Zhong, Wei Fan, Qiang Yang, Olivier Verscheure, and Jiangtao Ren. Cross validation framework to choose amongst models and datasets for transfer learning. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 547–562. Springer, 2010. 3

[29] Tinghui Zhou, Shubham Tulsiani, Weilun Sun, Jitendra Malik, and Alexei A Efros. View synthesis by appearance flow. In *ECCV*, 2016. 4

[30] Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. In *ICCV*, 2017. 5

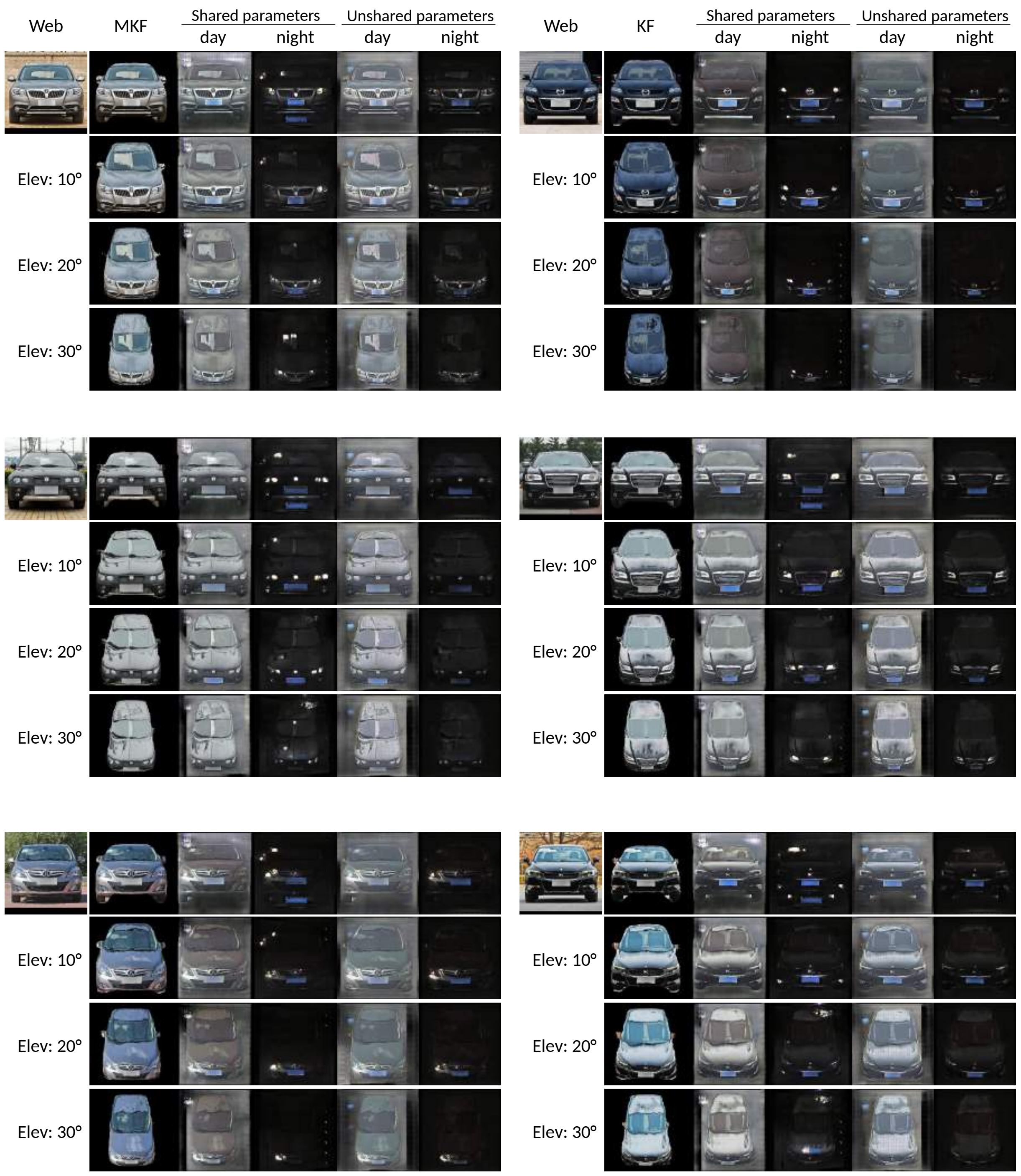


Figure S3: Visualization of synthesized images by photometric transformations using AC-CGANs with shared and unshared parameters.

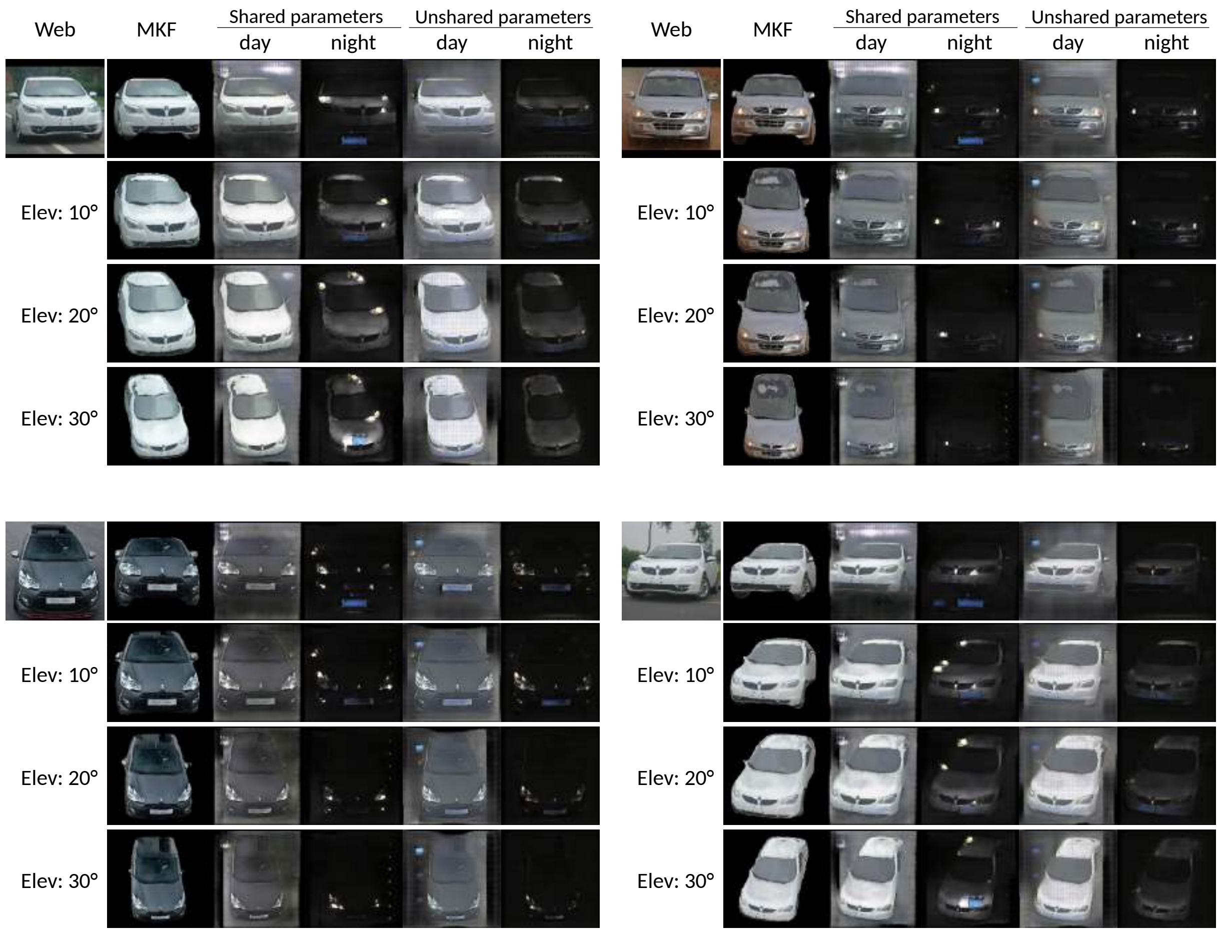
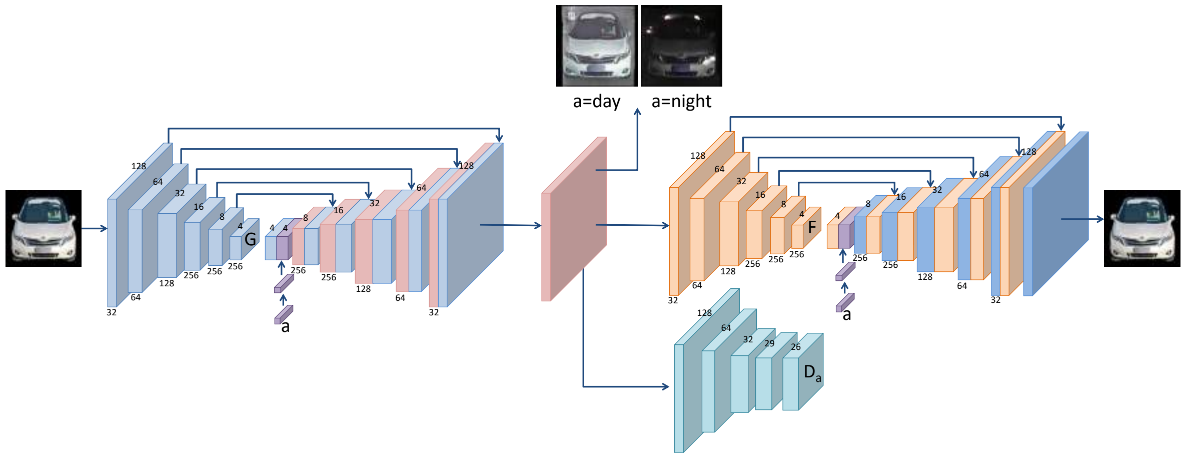
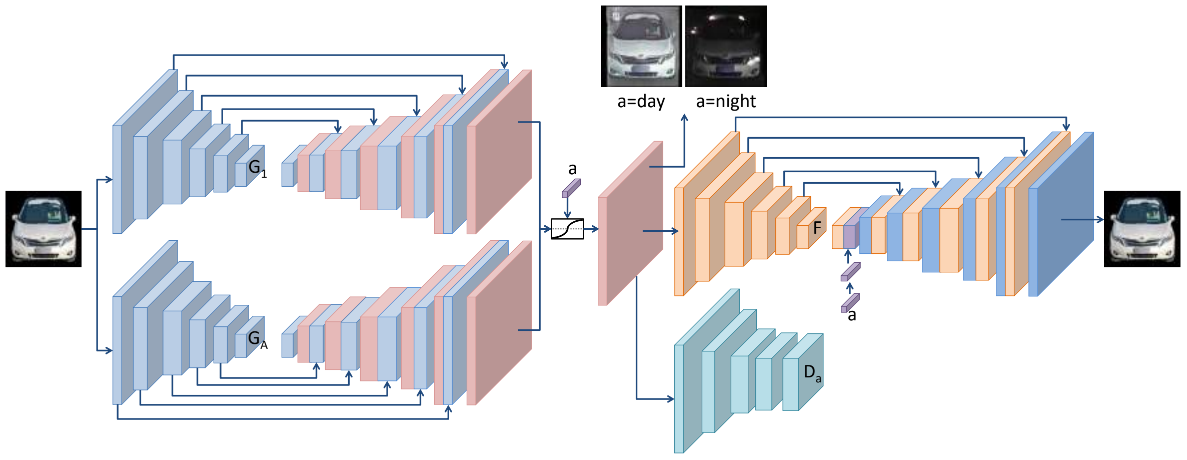


Figure S4: Visualization of synthesized images by photometric transformations using AC-CGANs with shared and unshared parameters for web images with different yaw angles from 0°.



(a) Attribute-conditioned CycleGAN



(b) Attribute-conditioned CycleGAN with unshared parameters

Figure S5: Networks architecture comparisons between AC-CGANs with shared and unshared parameters across generators and discriminators.