Adversarial Data Programming: Using GANs to Relax the Bottleneck of Curated Labeled Data

Arghya Pal, Vineeth N Balasubramanian
Department of Computer Science and Engineering
Indian Institute of Technology, Hyderabad, INDIA
{cs15resch11001,vineethnb}@iith.ac.in

Abstract

Paucity of large curated hand-labeled training data forms a major bottleneck in the deployment of machine learning models in computer vision and other fields. Recent work (Data Programming) has shown how distant supervision signals in the form of labeling functions can be used to obtain labels for given data in near-constant time. In this work, we present Adversarial Data Programming (ADP), which presents an adversarial methodology to generate data as well as a curated aggregated label, given a set of weak labeling functions. We validated our method on the MNIST, Fashion MNIST, CIFAR 10 and SVHN datasets, and it outperformed many state-of-the-art models. We conducted extensive experiments to study its usefulness, as well as showed how the proposed ADP framework can be used for transfer learning as well as multi-task learning, where data from two domains are generated simultaneously using the framework along with the label information. Our future work will involve understanding the theoretical implications of this new framework from a game-theoretic perspective, as well as explore the performance of the method on more complex datasets.

1. Introduction

Curated labeled data is a key building block of modern machine learning algorithms, and a driving force for deep neural network models. The large parameter space of deep models requires very large labeled datasets to build effective models that work in practice. However, this inherited dependency on large curated labeled data has become the major bottleneck of progress in the use of machine learning and deep learning in computer vision and other domains [41]. Creation of large scale hand-annotated datasets in every domain is a challenging task due to the requirement for extensive domain expertise, long hours of human labor and time - which collectively make the overall process expensive and time-consuming. Even when data annotation is carried out using crowdsourcing (e.g. Amazon Mechanical Turk), additional effort is required to measure the correctness (or goodness) of the obtained labels. We seek to address this problem in this work. In particular, we focus on automatically learning the parameters of a given joint image-label probability distribution (as provided in training image-label pairs) with a view to automatically create labeled datasets.

To achieve this objective, we exploit the use of distant supervision signals to generate labeled data. These distant supervision signals are provided to our framework as a set of weak labeling functions which represent domain knowledge or heuristics obtained from experts or crowd annotators. Writing a set of labeling functions (as we found in our experiments) is fairly easy and quick, and can then be used in our framework to generate data as well as associated labels. More interestingly, such labeling functions are often easily generalizable, thus allowing our framework to be extended to transfer learning and multi-task learning (discussed in Section 5). Figure 1 shows a few examples of our results to illustrate the overall idea.

In practice, labeling functions can be associated with two kinds of dependencies: (i) relative accuracies, which measure the correctness of the labeling functions w.r.t. the true class label; and (ii) inter-function dependencies that capture the relationships between the labeling functions with respect to the predicted class label. In this work, we have proposed a novel adversarial framework using Generative Adversarial Networks (GANs) that learns these dependencies along with the data distribution using a minmax game. Our GAN learns to generate a joint data-label distribution using a generator block, a discriminator block and a Labeling Functions Block (LFB), which contains another discriminator that helps in learning the two kinds of dependencies mentioned above. The overall architecture of the proposed ADP architecture is presented in Figure 2a.

Our broad idea of learning relative accuracies and inter-function dependencies of labeling functions is inspired by

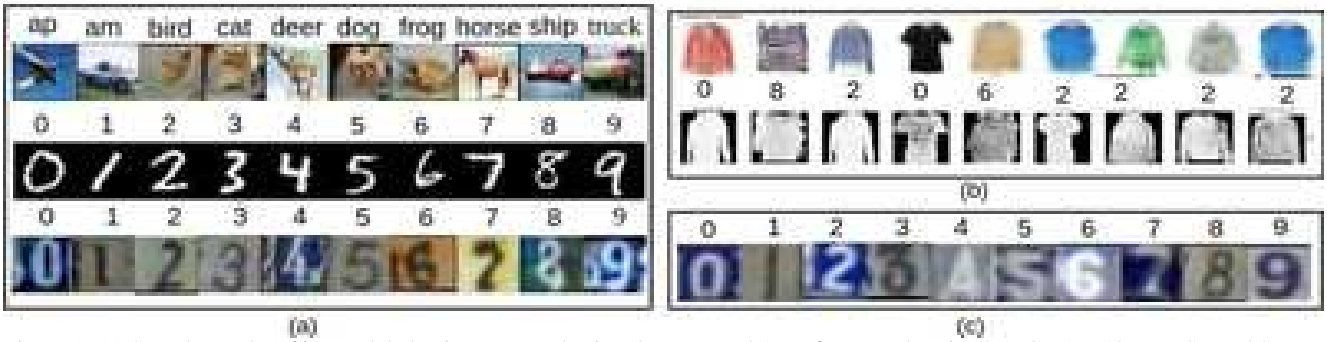

Figure 1: (a) Sample results of image-label pairs generated using the proposed ADP framework trained on CIFAR-10, MNIST and SVHN datasets (top to bottom respectively). Note that the label is generated by our model; (b) Demonstration of cross-domain multi-task learning using ADP , where the same model generates data from both Fashion MNIST and LookBook datasets (Section 5). Note that Fashion MNIST is grayscale while LookBook is color, and the model still generates both data effectively; (c) Demonstration of transfer learning of our ADP from MNIST dataset (source domain) to generate image-label pairs on the SVHN dataset (target domain).

the recently proposed Data Programming (DP) framework [36] (and hence, the name ADP), but our method is different in many ways: (i) DP is a strict conditional model (i.e. $P(\tilde{y}|\mathbf{x})$) that requires additional unlabeled data points even at test time, while our model is a joint distribution model, i.e. $P(\mathbf{x}, y)$; (ii) DP learns a generative model using Maximum Likelihood Estimation (MLE) and gradient descent to learn the relative accuracies of labeling functions. We however replace this approach with a GAN-based adversarial estimation of parameters. [11] and [42] provide insights on the advantage of using a GAN-based estimator over MLE to achieve a relatively quicker training time and good robustness on generated samples. (iii) In order to learn the statistical dependencies of labeling functions, we use an adversarial approach as an estimator (hence the second discriminator in Section 3.3.2) to replace the computationally expensive factor graph and Gibbs sampling techniques to update the gradient in each step.

As our outcomes of this work, we show how a set of low-quality, weak labeling functions can be used within a framework that models a joint data-label distribution to generate robust samples. We also show that this idea can be generalized quite easily to transfer learning and multi-task learning settings. To summarize:

- We propose a novel adversarial framework, ADP , to generate robust data-label pairs that be used to obtain datasets in domains that have very little data and thus save human labor and time.
- We show how an adversarial framework can be used to learn dependencies between weak labeling functions and thus provide high-fidelity aggregated labels along with generated data in a GAN setting.
- The proposed framework can also be used in a transfer learning setting where ADP can be trained on a source domain, and then finetuned on a target domain to then generate data-label pairs in the target domain.
- We also show the potential of this ADP framework to generate cross-domain data in a multi-task setting, where images from two domains are generated simultaneously by the model along with the labels.

2. Related Work

Data augmentation seems a natural answer to the scarcity of curated hand-labeled training data. However, heuristic data augmentation techniques like [15] and [19] use a limited form of class-preserving image transformations. Interpolation-based methods proposed in [13] and class-conditional models of diffeomorphisms proposed in [20] interpolate between nearest-neighbor labeled data points.

In this work, we choose to use a more intuitive way of creating labeled data by learning a joint distribution model. Learning a joint data-label distribution using generative models such as [14], [18], and [28] is non-trivial, since the label often requires domain knowledge and not directly inferrably from data. Our proposed model hence uses distant supervision signals (in the form of labeling functions) to *generate* novel labeled data points. Distant supervision signals such as labeling functions are cheaper than manual annotation of each data point, and has been successfully used in recent methods such as [36]. Ratner et al. proposed a generative model in [36] that uses a fixed number of user-defined labeling functions to programatically generate synthetic labels for data in near-constant time. DP outperformed number of approaches such as multiple-instance learning [38], co-training [4], crowdsourcing [17], or ensemble based weak-learner method like boosting [40], thus reinforcing our choice in this work. The popular SMOTE algorithm [7] performs oversampling to reduce class imbalance and augment the given data. Unfortunately, all of these methods vastly depend on hand-tuned parameters, the order of geometric transformations, the optimal value of transfor-

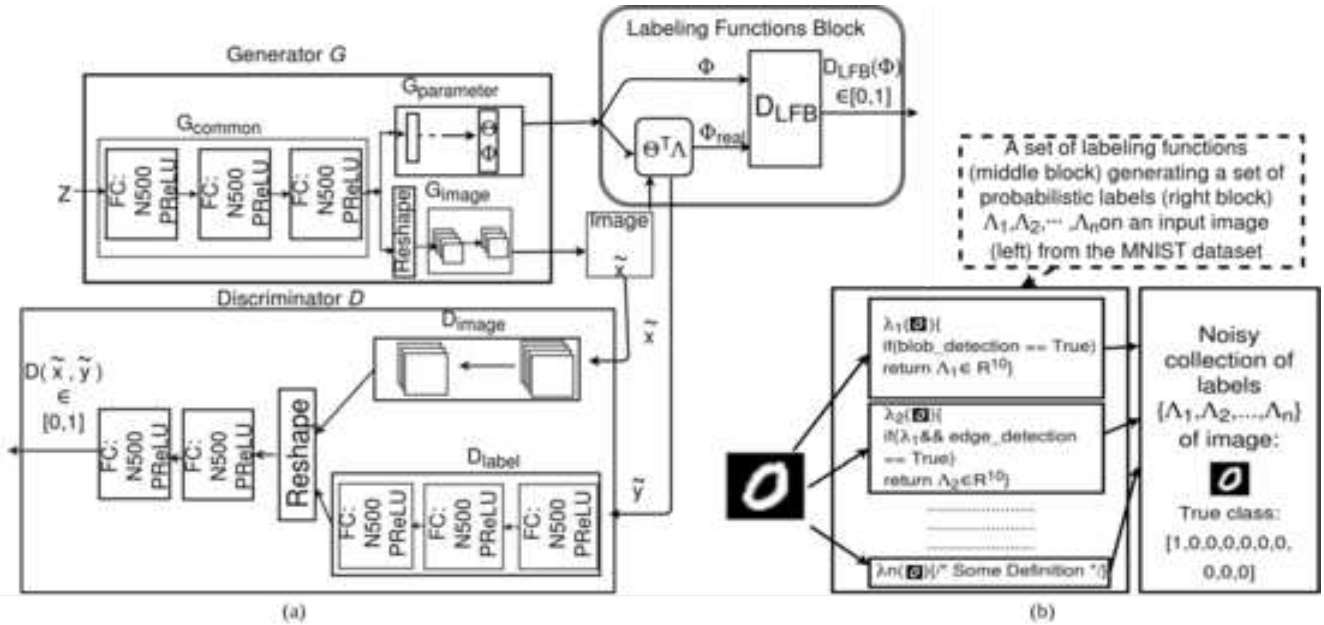

Figure 2: (a) Overall architecture of the Adversarial Data Programming (ADP) framework; (b) Example of a set of labeling functions

mation parameters, etc. as studied in [37], [10] and [15]). Alfonseca et al. [1] generated additional training data using hierarchical topic models for weak supervision. Heuristics for distant supervision are also proposed in [6], but this method does not model the inherent noise associated with such heuristics. Structure learning [43], [37] also exploits the use of distant supervision signals for generating labels, but as described in Section 1, these methods like [36] require unlabeled test data to generate a labeled dataset. Additionally, [36], [37] and [43] are computationally expensive due to the use of Gibbs sampling in MLE.

We instead use an adversarial approach to learn the joint distribution by weighting a set of domain-specific label functions using a Generative Adversarial Network (GAN). GAN [18] approximates the real data distribution by optimizing a minmax objective function and thus generates novel out-of-sample data points. Broadly, GAN can be viewed in terms of three manifestations: (i) GANs can be trained to sample from a marginal distribution $P(\mathbf{x})$ [12], [35][2], where $\mathbf{x}$ refers to data. (ii) Recent efforts in literature such as Conditional GAN [31], Auxiliary Classifier GAN [34] and InfoGAN [9] show training of GANs conditioned on class labels y to thus sample from a conditional distribution, i.e. $P(\mathbf{x}|y)$. Other state-of-the-art models with similar objectives have exploited other modalities for the same purpose; for example, Zhang et al [49] propose a GAN conditioned on images, while Hu et al [21] propose a GAN conditioned on text. (iii) There have been a few very recent efforts [46], [51] and [22], which attempt to train GANs to sample from a joint distribution. For example, CoGAN [29] introduces a parameter-sharing approach to learn an unpaired joint distribution between two domains, while TripleGAN [27] brings together a classifier along with the discriminator and generator which helps in a semi-supervised setting. In this work, we propose a novel idea to instead use distant supervision signals to accomplish learning the joint distribution of labeled images. We now describe the proposed methodology.

3. Adversarial Data Programming (ADP): Methodology

Our central aim in this work is to learn parameters of a probabilistic model:

$$P(\mathbf{x}, y) \tag{1}$$

that captures the joint distribution over the data $\mathbf{x}$ and the corresponding labels y, thus allowing us to generate out-of-sample data points along with their corresponding labels (we focus on images in the rest of this paper).

While recent efforts such as [29] and [16] have considered complementary objectives, they largely focused on learning joint probability distributions in cross-domain understanding settings. In this work, we focus on learning the joint image-label probability distribution with a view to automatically create labeled datasets, by exploiting the use of distant supervision signals to generate labeled data. To the best of our knowledge, this is the first such work that invokes distant supervision while learning the joint distribution $P(\mathbf{x}, y)$, so as to generate labeled data points at scale from $P(\mathbf{x}, y)$. Besides, automatic generation of labels for data based on training data-label pairs is non-trivial, and often does not work directly. Distant supervision provides

us a mechanism to achieve this challenging goal. We encode distant supervision signals as a set of (weak) definitions by annotators using which unlabeled data points can be labeled. These definitions can be harvested from knowledge bases, domain heuristics, ontologies, rules-of-thumb, educated guesses, decisions of weak classifiers or obtained using crowdsourcing. Many application domains have such distant supervision available in different means through domain knowledge or heuristics, which can be leveraged in the proposed framework. We provide examples in Section 4 when we describe our experiments.

We encapsulate all available distant supervision signals, henceforth called *labeling functions*, in a unified abstract container called **L**abeling **F**unctions **B**lock (LFB, see Figure 2a). Let LFB comprise of n labeling functions $\lambda_1, \lambda_2, \cdots, \lambda_n$, where each labeling function is a mapping:

$$\lambda_i : \mathbf{x}_j \rightarrow \Lambda_{ij} \tag{2}$$

that maps a data point $\mathbf{x}_j$ to a m-dimensional probabilistic label vector, $\Lambda_{ij} \in \mathbb{R}^m$, where m is the number of class labels with $\sum_m \Lambda_{ij} = 1$ and $0 \leq \Lambda_{ij}^k \leq 1$ for each $k \in \{1, \cdots, m\}$. For example, $\mathbf{x}_j$ could be thought of as an image from the MNIST dataset, and $\Lambda_{ij} \in \mathbb{R}^{10}$ would be the corresponding label vector when the labeling function λ_i is applied to $\mathbf{x}_j$. Λ_{ij}, for instance, could be the one-hot 10-dimensional class vector, see Figure 2b.

We characterize the set of labeling functions, $\{\lambda_i, i = 1, \cdots, n\}$, with two kinds of dependencies: (i) *relative accuracies* of the labeling functions with respect to the true class label of a given data point; and (ii) *inter-function dependencies* that capture the relationships between the labeling functions with respect to the predicted class label. To obtain a final label y for a given data point $\mathbf{x}$ using the LFB, we use two different sets of parameters, Θ and Φ to capture each of these dependencies between the labeling functions. We, hence, denote the Labeling Function Block (LFB) as:

$$LFB_{\lambda,\Theta,\Phi} : \mathbf{x}_j \rightarrow \Lambda_j \tag{3}$$

i.e. given a set of labeling functions λ, a set of parameters capturing the relative accuracy-based dependencies between the labeling functions, Θ, and a second set of parameters capturing inter-label dependencies, Φ, LFB provides a probabilistic label vector, Λ_j, for a given data input $\mathbf{x}_j$.

The joint distribution we seek to model in this work (Equation 1) hence becomes:

$$P(\mathbf{x}, LFB_{\lambda,\Theta,\Phi}(\mathbf{x})) \tag{4}$$

In the rest of this section, we show how we can learn the parameters of the above distribution modeling image-label pairs using an adversarial framework with a high degree of label fidelity. We use Generative Adversarial Networks (GANs) to model the joint distribution in Equation 4. In particular, we provide a mechanism to integrate the LFB (Equation 3) into the GAN framework, and show how Θ and Φ can be learned through the framework itself. Our adversarial loss function is given by:

$$\min \max L(G, D) = \mathbb{E}_{(\mathbf{x},y)\sim P_{real}(\mathbf{x},y)} \log(D(\mathbf{x}, y)) + \mathbb{E}_{(\tilde{\mathbf{x}},\Lambda)\sim P_{fake}(z)} \log(1 - D(\tilde{\mathbf{x}}, \Lambda)) \tag{5}$$

where G is the generator module and D is the discriminator module. The overall architecture of the proposed ADP framework is shown in Figure 2a.

This approach has a few advantages: (i) labeling functions (which can even be just loosely defined) are cheaper to obtain than collecting labels for a large dataset; (ii) labeling functions can help bring domain knowledge into such generative models; (iii) labeling functions act as an implicit regularizer in the label space, thus allowing good generalization; (iv) with a small fine-tuning, labeling functions can be easily re-purposed for new domains (*transfer learning*), as we describe later in this paper.

The ADP architecture is designed to learn the parameters required to model the joint distribution in Equation 4, and thus generate out-of-sample image-label pairs. This architecture is broadly divided into three modules: the generator, discriminator and the LFB. We now describe each of these modules individually.

3.1. The ADP - Generator

Given a noise input z and a set of labeling functions λ, the generator G outputs an image $\mathbf{x}$ and the parameters Θ and Φ, the dependencies between the labeling functions described earlier. In particular, G consists of three blocks: G_{common}, G_{image} and $G_{parameter}$, as shown in Figure 2a. G_{common} captures the common high-level semantic relationships between the data and the label space, and is comprised only of fully connected (FC) layers. The output of G_{common} forks into two branches: G_{image} and $G_{parameter}$, where G_{image} generates the image $\tilde{\mathbf{x}}$, and $G_{parameter}$ generates the parameters (Θ, Φ). While $G_{parameter}$ uses FC layers, G_{image} uses Fully Convolutional (FCONV) layers to generate the image (more details in Section 4). Thus, the generator G outputs $(\mathbf{x}, \Theta, \Phi)$ given input $z \sim \mathcal{N}(0, I)$, the standard normal distribution.

3.2. The ADP - Discriminator

The discriminator D of ADP estimates the likelihood of an image-label input pair being drawn from the real distribution obtained from training data. D takes a batch of image-label pairs as input and maps that to a probability score to estimate the aforementioned likelihood of the image-label pair. To accomplish this, D has two branches: D_{image} and D_{label} (shown in the Discriminator block in

Figure 2a). These two branches are not coupled in the initial layers, so as to separately extract required low-level features. The branches share weights in later layers to extract joint semantic features that help D classify correctly if an image-label pair is *fake* or *real*. We hence expand our objective function from Equation 5 to the following:

$$\begin{aligned}\min\max L(G, D_{image}, D_{label}) = \\ \mathbb{E}_{(\mathbf{x},y)\sim P_{real}(\mathbf{x},y)} \log(D_{image}(\mathbf{x})) \\ + \quad \mathbb{E}_{z\sim\mathcal{N}(0,I)} \log(1 - D_{image}(G_{image}(z))) \\ + \quad \mathbb{E}_{(\mathbf{x},y)\sim P_{real}(\mathbf{x},y)} \log(D_{label}(y)) \\ + \quad \mathbb{E}_{z\sim\mathcal{N}(0,I)} \log(1 - D_{label}(LFB(G(z))))\end{aligned} \tag{6}$$

3.3. The ADP - Labeling Function Block

This is a critical module of the proposed ADP framework. Our initial work revealed that a simple weighted (linear or non-linear) sum of the labeling functions do not perform well in generating out-of-sample image-label pairs. We hence used a separate adversarial methodology within this block to learn the dependencies. We describe the components of the LFB below.

3.3.1 Relative Accuracies of Labeling Functions

The output, Θ, of the $G_{parameter}$ block in the ADP-Generator G provides the relative accuracies of the labeling functions. Given the image output generated by G_{image}: $\tilde{\mathbf{x}}$, the labeling functions $\{\lambda_1, \cdots, \lambda_n\}$, and the probabilistic label vectors $\{\Lambda_i, i = 1, \cdots, n\}$ obtained using the labeling functions (as in Eqn 2), we define the aggregated final label as:

$$\tilde{y} = \sum_{i=1}^{n} \tilde{\theta}_i \Lambda_i = \tilde{\Theta} \cdot \Lambda \tag{7}$$

where $\tilde{\theta}_i$ is the normalized version of θ_i, i.e. $\tilde{\theta}_i = \frac{\theta_i}{\sum_{k=1}^{n} \theta_k}$. The aggregated label, $\tilde{y}$, is provided as an output of the LFB.

3.3.2 Inter-function Dependencies

Our empirical studies with considering only relative accuracies of labeling functions as a weighting mechanism led to *mode collapse* in the joint distribution space, a well-understood problem in GANs. Our preliminary empirical studies demonstrated mode collapse in the joint distribution space - either images of same class with different labels, or images of different classes with same label were generated (please see Fig 3). The rationale behind taking two discriminators is to penalize the missing modes. Related literature [36] shows that inter-functional dependencies act as an implicit regularizer in the label space. We hence introduced an adversarial mechanism inside the LFB to influence the final relative accuracies, $\tilde{\theta}$, using the inter-function dependencies between the labeling functions. D_{LFB}, a discriminator inside LFB, receives two inputs: Φ, which is output by $G_{parameter}$, and Φ_{real}, which is obtained from Θ using the procedure described in Algorithm 1.

Algorithm 1: Procedure to compute Φ_{real}

Input: Labeling functions $\{\lambda_1, \cdots, \lambda_n\}$, Relative accuracies $\theta_1, \cdots, \theta_n$, Output probability vectors of labeling functions $\Lambda_1, \cdots, \Lambda_n$

Output: Φ_{real}

Set $\Phi_{real} = \mathrm{I}(n, n)$;
`/* I = Identity Matrix */`
for $i = 1$ *to* n **do**
  `/* For each labeling function */`
  for $j = i + 1$ *to* n **do**
    `/* For each other labeling function */`
    `/* If one-hot encoding of the outputs of two functions match, increment` (i, j)`th entry in` Φ_{real} `by 1 */`
    $\Phi_{real}(i, j) = \Phi_{real}(i, j) + \text{OneHot}(\theta_i \Lambda_i) \cdot \text{OneHot}(\theta_j \Lambda_j)$;
  end
end
for $p = 1$ *to* n **do**
  $\Phi_{real}(\text{p}, .) = \frac{\Phi_{real}(p,.)}{\sum_{u=1}^{n} \Phi_{real}(p,u)}$;
end
Set $\Phi_{real} = \Phi_{real} + \Phi_{real}^{T} - \text{diag}(\Phi_{real})$
`/* Complete matrix using symmetry */`

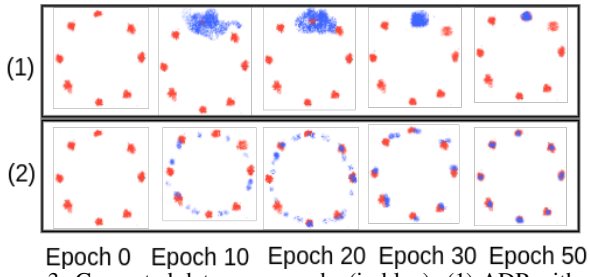

Figure 3: Generated data over epochs (in blue): (1) ADP without inter-functional dependencies, (2) ADP with inter-functional dependencies. Data from true distribution (8 different Gaussians) in red.

Algorithm 1 computes a matrix of interdependencies between the labeling functions, Φ_{real}, by looking at the one-hot encodings of their predicted label vectors. If the one-hot encodings match for a given data input, we increase the count of their correlation by one, and compute this matrix across a particular mini-batch of data points under consideration. The counts are then normalized row-wise to obtain Φ_{real}. The task of the discriminator is to recognize the computed interdependencies as real, and the Φ generated through the network in $G_{parameter}$ as fake. The gradient

backpropagated through this discriminator to the G block is critical as a regularizer in learning a better Θ, which is finally used to weight the labeling functions (as in Section 3.3.1). Combining the gradient information from D_{LFB} along with D, penalizes missing modes and helps G to generate more variety in the samples. The objective function of our second adversarial module is hence:

$$\begin{aligned} &\min\max L(D_{LFB}, G) = \\ &+ \quad \mathbb{E}_{z\sim\mathcal{N}(0,I)} \log(D_{LFB}(\Phi_{real}(z))) \\ &+ \quad \mathbb{E}_{z\sim\mathcal{N}(0,I)} \log(1 - D_{LFB}(\Phi(z))) \end{aligned} \tag{8}$$

where Φ_{real} and Φ are obtained from $G_{parameter}(z)$ as described above. More details of the LFB are provided in implementation details in Section 4. The overall architecture of ADP (Figure 2a) is trained using end-to-end backpropagation with gradients from both discriminators, D and D_{LFB}, influencing the weights learned inside the generator G. Mini-batches of image-label pairs from a given training distribution are provided as input to ADP , and Stochastic Gradient Descent (SGD) is used to learn the parameters of the model. At the end of training, we define the aggregated final label as:

$$\tilde{y} = \tilde{\Theta} \cdot \Phi^T \cdot \Lambda \tag{9}$$

the samples $(\tilde{\mathbf{x}}, \tilde{y})$ generated using the G and LFB modules thus provide samples from the desired joint distribution (Eqn 1) modeled using the framework.

Type	Labeling Functions used
Heuristic	Presence of long edges (vertical or horizontal) [30]; Image histogram
Image Processing based	Bag-of-feature [39]; Haar wavelet [8]; Discrete-continuous ADM [25]; Compressive sensing [50]
Deep Learning based	Convolution kernels from last conv layer (before fully connected layers) of LeNet

Table 1: Labeling functions used for MNIST and SVHN datasets, both of which represent the digit recognition task

4. Experiments and Results

4.1. Datasets

We validated the ADP framework on standard datasets: MNIST [26], Fashion MNIST [45], SVHN [33], and CIFAR-10 [23] with no additional pre-processing on the datasets.[1].

4.2. Labeling Functions

Labeling functions form a critical element of ADP , and we used different cues from state-of-the-art algorithms to help obtain labeling functions for our experiments. Table 1 shows the labeling functions we used for our experiments on MNIST and SVHN (digit recognition problems), and Table 2 shows the functions used for CIFAR and Fashion-MNIST. We categorized labeling functions as: (i) Heuristic; (ii) Image processing-based; and (iii) Deep learning-based labeling functions (as in Tables 1 and 2). Table 3 presents the statistics of the number of labeling functions used for each of the considered datasets (the empirical study that motivated these choices is presented in Section 5). In this work, for each labeling function, a simple threshold rule on the L_2-norm of the aforementioned features is used, where the threshold is obtained empirically as the mean of the L_2-norms of a randomly chosen subset, which is α-trimmed to remove outliers. More examples of labeling functions and ablation studies on their usefulness are presented in the Supplementary Section.

[1]Code available at https://github.com/ArghyaPal/Adversarial-Data-Programming

Type	Labeling Functions used
Heuristic	PatchMatch [3]; Blob Detection; Presence of edges [1]; Textons; Image histogram
Image Processing based	Global descriptor (GIST-based) [32]; Local descriptor (SIFT-based) [48]; Bag-of-visual-words; Histogram of Oriented Gradient (HOG)-based: HoGgles [44]
Deep Learning based	Convolution kernels from last conv layer (before fully connected layers) of (ImageNet) pre-trained AlexNet

Table 2: Labeling functions used for CIFAR 10 and Fashion MNIST datasets

	Heuristic	Image Processing	Deep Learning
MNIST	43	10	1
Fashion-MNIST	50	6	1
SVHN	43	10	2
CIFAR 10	46	18	2

Table 3: Number of labeling functions used for different datasets

4.3. Implementation Details

G_{common} has 3 dense fully connected layers (FC) (128 nodes per layer) with batch-normalization. G_{image} continues with fractional length convolutional layers, similar to [29] (FCONV: 1024 nodes per layer, Kernel size: 4×4, Stride: 1, followed by batch-normalization and Parameterized ReLU), and generates image $\mathbf{x}$. $G_{parameter}$ uses FC layers and generates Θ, Φ. The discriminator network D follows the "in-plane rotation" network of [29]. D_{label} is a stack of FC layers. Both D_{image} and D_{LFB} have 2 FCONV layers followed by FC layers. We trained the complete model with mini-batch Stochastic Gradient Descent (SGD) using a batch size of 128, learning rate of 0.0001, momentum factor of 0.5 and Adam as an optimizer.

4.4. Comparison with State-of-the-Art Models

Qualitative Results: We compared our method against other generative methods that allow generation of data along with a label: Conditional GAN or CGAN [18], ACGAN [34], InfoGAN [9], CoGAN [29] and TripleGAN

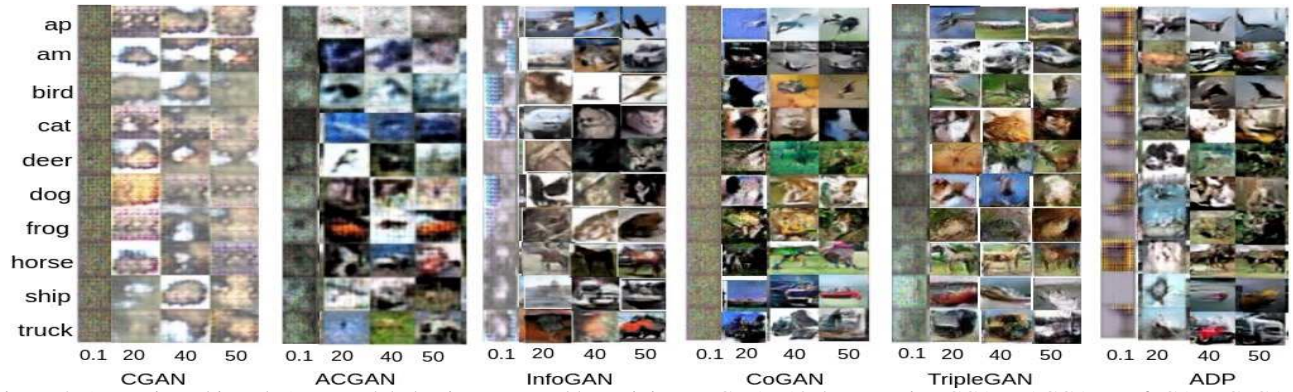

Figure 4: (Best viewed in color) Image-label pairs generated by training on CIFAR10 dataset using CGAN, ACGAN, InfoGAN, CoGAN, TripleGAN and our method, ADP . For a given model, the columns of images represents generations after 0.1k, 20k, 40k, 50k epochs, and the rows correspond to the associated class label. 'ap' stands for airplane, and 'am' stands for the automobile class of CIFAR 10 dataset. Note the clarity of generations of the proposed method.

Dataset	**Image Quality**					**Image-Label Correspondence**				
	ACGAN	CGAN	InfoGAN	TripleGAN	ADP	ACGAN	CGAN	InfoGAN	TripleGAN	ADP
MNIST	9.02 ± 0.1	9.11 ± 0.2	9.54 ± 0.3	$\mathbf{9.6 \pm 0.4}$	9.46 ± 0.3	8.27 ± 0.3	9.11 ± 0.2	9.78 ± 0.2	9.6 ± 0.4	$\mathbf{9.92 \pm 0.1}$
FMNIST	9.32 ± 0.4	8.89 ± 0.3	9.10 ± 0.3	9.2 ± 0.2	$\mathbf{9.33 \pm 0.6}$	8.8 ± 0.1	8.89 ± 0.3	9.27 ± 0.4	9.2 ± 0.2	$\mathbf{9.93 \pm 0.1}$
SVHN	5.3 ± 0.2	4.91 ± 0.5	7.71 ± 0.1	8.6 ± 0.3	$\mathbf{8.86 \pm 0.3}$	8.53 ± 0.3	8.91 ± 0.0	9.08 ± 0.1	$\mathbf{9.75 \pm 0.2}$	9.72 ± 0.3
CIFAR10	4.17 ± 0.1	4.36 ± 0.2	6.23 ± 0.2	$\mathbf{8.5 \pm 0.1}$	8.27 ± 0.3	7.27 ± 0.1	8.62 ± 0.2	$\mathbf{9.72 \pm 0.1}$	9.68 ± 0.3	9.49 ± 0.5

Table 4: Human Turing Test for image quality and image-label correspondence (Section 4.4, higher the better). Note that the proposed method, ADP performs the best in most cases, and is a close second when TripleGAN wins.

Figure 5: Transfer learning from MNIST to SVHN dataset. Digits within parentheses indicate true label, while the other is the label generated using our method (Section 5, Transfer Learning)

No of Labeling Functions	MNIST	F-MNIST	CIFAR10	SVHN
3	70.23%	81.02%	87.39%	83.82%
25	*20.32%*	*30.53%*	*42.31%*	*38.30%*
40	1.40%	6.81%	19.93%	16.62%
50	**1.33%**	4.92%	18.93%	13.05%
55	1.34%	**4.80%**	**18.45%**	**12.83%**
65	1.31%	4.73%	18.43%	12.82%

Table 5: Performance of ADP when number of labeling functions is varied (Section 5, Optimal Number of Labeling Functions).

[27]. We changed the use case setup of these methods to generate data-label pairs as required. Results for CIFAR10 are shown in Figure 4, and, results for other datasets are shown in the Supplementary Section. While some of the aforementioned methods (such as CGAN and InfoGAN) generate images conditioned on a given label (and hence require a label to be provided as input), the label is provided by the model in our case.

Quantitative Results: We considered three evaluation metrics for studying the performance of our method quantitatively: (i) *Human Turing Test (HTT):* This metric studies how hard it is for a human annotator to tell the difference between real and generated samples. We asked 40 subjects to evaluate image quality and image-label correspondence (Table 4) on a scale of 10, given 50 random image-label samples from the generated pool for each method considered. Table 4 shows consistently good performance of ADP over other methods, especially in image-label correspondence, which is the focus of this work; (ii) *Inception Score:* The inception score, as used in [29] and [27], for the CIFAR 10 dataset is shown in Figure 7. The figure shows that ADP and TripleGAN perform significantly better than the rest of the methods (more results on other datasets included in the Supplementary Section). We also used a *Parzen window based evaluation* metric, and these results are included in the Supplementary Section.

(Training Data, Test Data)	Epochs						
	5k	10k	15k	20k	30k	40k	50k
(Real data-50K, Real data-10K)	9.83	7.3	7.12	6.3	6.1	4.3	4.19
(ADP data-50K, Real data-10K)	9.32	8.9	8.13	7.0	6.75	5.53	5.0
(Real data-50K, ADP data-10K)	9.67	9.4	7.92	7.3	6.81	6.18	5.6
(ADP-25K + Real data-25K, Real data-10K)	8.5	6.6	6.21	5.7	5.5	4.83	3.5
(ADP-50K + Real data-50K, Real data-10K)	**7.71**	**6.3**	**6.0**	**5.34**	**3.1**	**2.92**	**2.71**

Table 6: Test cross-entropy loss of ResNet-56 on CIFAR-10 dataset. (Real data-50K, Real data-10K) = standard dataset; ADP = our method; 10/25/50K = the number of data points used in thousands. In ADP-25K + Real data-25K, class ratios were maintained as in the original dataset.

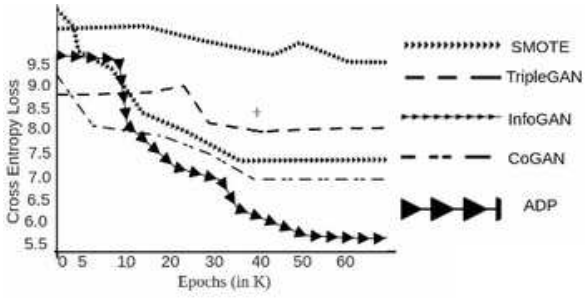

Figure 6: Classification performance of a pretrained ResNet model on image-label pairs generated by various models trained on CIFAR 10

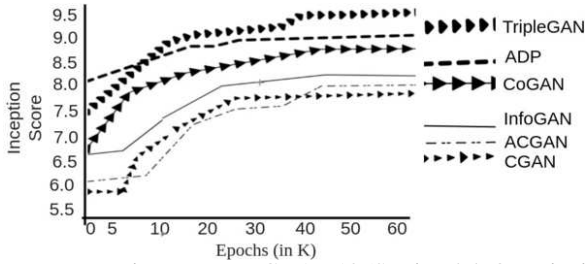

Figure 7: Inception scores on CIFAR 10 (Section 4.4, Quantitative Analysis)

Classification Performance: To study the usefulness of the generated image-label pairs, we studied the classification cross-entropy loss of: (i) pretrained ResNet-56 model on the image-label pairs generated by our ADP at test time and compared our method against TripleGAN, InfoGAN, CoGAN as well as the popular oversampling technique, SMOTE [7] (see Figure 6), and, (ii) We trained a ResNet-56 model on the image-label pairs generated by our ADP and tested on CIFAR-10 dataset under different settings, and the results are shown in Table 6.

5. Discussion and Analysis

Optimal Number of Labeling Functions: We studied the performance of ADP when the number of labeling functions is varied to understand the impact of this parameter on the performance. We studied the test cross-entropy error of a pretrained ResNet model with image-label pairs generated by ADP, trained using different number of labeling functions. Table 5 shows our results, suggesting that 50-55 labeling functions provides the best performance, depending on the dataset. This justifies our choice of number of labeling functions in Table 3.

Transfer Learning: The use of distant supervision signals such as labeling functions (which can often be generic) allows us to extend the proposed ADP model to a transfer learning setting. In this setup, we trained ADP initially on a source dataset and then finetuned the model to a target dataset, with very limited training. In particular, we first

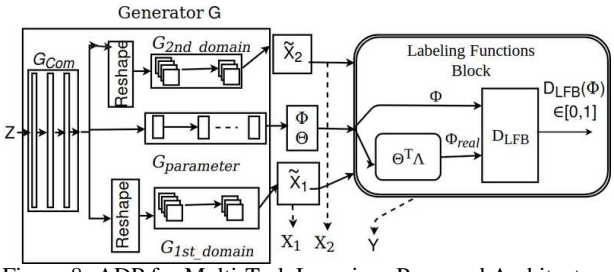

Figure 8: ADP for Multi-Task Learning: Proposed Architecture

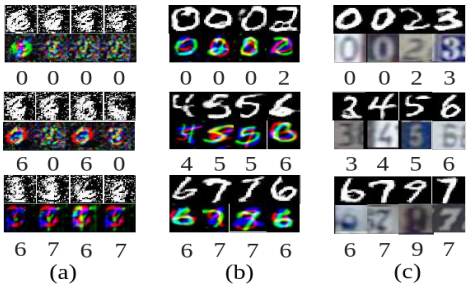

Figure 9: Results of ADP - Multi-Task Learning on MNIST (black and white) and SVHN (RGB) datasets: (a) Intial epochs (b) Middle of training (c) generations at higher epochs

trained ADP on the MNIST dataset, and subsequently finetuned the G_{image} branch alone with the SVHN dataset. We note that the weights of G_{common}, $G_{parameter}$ and D_{LFB} are unaltered. The final finetuned model is then used to generate image-label pairs (which we hypothesize will look similar to SVHN). Figure 1 shows encouraging results of our experiments in this regard.

Multi-task Joint Distribution Learning: Learning a cross-domain joint distribution from heterogeneous domains is a challenging task. We show that the proposed ADP method can be used to achieve this, by modifying its architecture as shown in Figure 8, to simultaneously generate data from two different domains. We study this architecture on the MNIST and SVHN datasets, and show the promising results of our experiments in Figure 9. The LFB acts as a regularizer and maintains the correlations between the domains in this case. More results on other datasets - in particular, LookBook and Fashion MNIST - are included in the Supplementary Section as well as Figure 1.

6. Conclusions

Paucity of large curated hand-labeled training data for every domain-of-interest forms a major bottleneck in deploying machine learning methods in practice and standard data augmentation techniques are often limited in their scope. Our proposed ADP framework learns the joint data-label distribution effectively using a set of weakly defined labeling functions. The method shows promise on standard datasets, as well as in transfer learning and multi-task learning.

References

[1] E. Alfonseca, K. Filippova, J.-Y. Delort, and G. Garrido. Pattern learning for relation extraction with a hierarchical topic model. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Short Papers-Volume 2*, pages 54–59. Association for Computational Linguistics, 2012.

[2] M. Arjovsky, S. Chintala, and L. Bottou. Wasserstein gan. *arXiv preprint arXiv:1701.07875*, 2017.

[3] C. Barnes, D. B. Goldman, E. Shechtman, and A. Finkelstein. The patchmatch randomized matching algorithm for image manipulation. *Communications of the ACM*, 54(11):103–110, 2011.

[4] A. Blum and T. Mitchell. Combining labeled and unlabeled data with co-training. In *Proceedings of the eleventh annual conference on Computational learning theory*, pages 92–100. ACM, 1998.

[5] O. Breuleux, Y. Bengio, and P. Vincent. Unlearning for better mixing.

[6] R. Bunescu and R. Mooney. Learning to extract relations from the web using minimal supervision. In *ACL*, 2007.

[7] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer. Smote: synthetic minority over-sampling technique. *Journal of artificial intelligence research*, 16:321–357, 2002.

[8] X. Chen, X. Cheng, and S. Mallat. Unsupervised deep haar scattering on graphs. In *Advances in Neural Information Processing Systems*, pages 1709–1717, 2014.

[9] X. Chen, Y. Duan, R. Houthooft, J. Schulman, I. Sutskever, and P. Abbeel. Infogan: Interpretable representation learning by information maximizing generative adversarial nets. In *Advances in Neural Information Processing Systems*, pages 2172–2180, 2016.

[10] D. Ciresan, U. Meier, L. Gambardella, and J. Schmidhuber. Deep big simple neural nets excel on handwritten digit recognition [ ]. 2010. *: http://arxiv. org/pdf/1003.0358, . . .(03.07. 2014)*.

[11] I. Danihelka, B. Lakshminarayanan, B. Uria, D. Wierstra, and P. Dayan. Comparison of maximum likelihood and gan-based training of real nvps. *arXiv preprint arXiv:1705.05263*, 2017.

[12] E. L. Denton, S. Chintala, R. Fergus, et al. Deep generative image models using a laplacian pyramid of adversarial networks. In *Advances in neural information processing systems*, pages 1486–1494, 2015.

[13] T. DeVries and G. W. Taylor. Dataset augmentation in feature space. *arXiv preprint arXiv:1702.05538*, 2017.

[14] C. Doersch. Tutorial on variational autoencoders. *arXiv preprint arXiv:1606.05908*, 2016.

[15] A. Dosovitskiy, P. Fischer, J. T. Springenberg, M. Riedmiller, and T. Brox. Discriminative unsupervised feature learning with exemplar convolutional neural networks. *IEEE transactions on pattern analysis and machine intelligence*, 38(9):1734–1747, 2016.

[16] Z. Gan, L. Chen, W. Wang, Y. Pu, Y. Zhang, H. Liu, C. Li, and L. Carin. Triangle generative adversarial networks. *arXiv preprint arXiv:1709.06548*, 2017.

[17] H. Gao, G. Barbier, and R. Goolsby. Harnessing the crowdsourcing power of social media for disaster relief. *IEEE Intelligent Systems*, 26(3):10–14, 2011.

[18] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative adversarial nets. In *Advances in neural information processing systems*, pages 2672–2680, 2014.

[19] B. Graham. Fractional max-pooling. *arXiv preprint arXiv:1412.6071*, 2014.

[20] S. Hauberg, O. Freifeld, A. B. L. Larsen, J. Fisher, and L. Hansen. Dreaming more data: Class-dependent distributions over diffeomorphisms for learned data augmentation. In *Artificial Intelligence and Statistics*, pages 342–350, 2016.

[21] Z. Hu, Z. Yang, X. Liang, R. Salakhutdinov, and E. P. Xing. Controllable text generation. *arXiv preprint arXiv:1703.00955*, 2017.

[22] T. Kim, M. Cha, H. Kim, J. Lee, and J. Kim. Learning to discover cross-domain relations with generative adversarial networks. *arXiv preprint arXiv:1703.05192*, 2017.

[23] A. Krizhevsky and G. Hinton. Learning multiple layers of features from tiny images. 2009.

[24] V. V. Kumar, A. Srikrishna, B. R. Babu, and M. R. Mani. Classification and recognition of handwritten digits by using mathematical morphology. *Sadhana*, 35(4):419–426, 2010.

[25] E. Laude, J.-H. Lange, F. Schmidt, B. Andres, and D. Cremers. Discrete-continuous splitting for weakly supervised learning. *arXiv preprint arXiv:1705.05020*, 2017.

[26] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[27] C. Li, K. Xu, J. Zhu, and B. Zhang. Triple generative adversarial nets. *arXiv preprint arXiv:1703.02291*, 2017.

[28] Y. Li, K. Swersky, and R. Zemel. Generative moment matching networks. In *Proceedings of the 32nd International Conference on Machine Learning (ICML-15)*, pages 1718–1727, 2015.

[29] M. Liu, O. Tuzel, and A. Sullivan. Coupled generative adversarial nets. 2016.

[30] S. Mandal, S. Sur, A. Dan, and P. Bhowmick. Handwritten bangla character recognition in machine-printed forms using gradient information and haar wavelet. In *Image Information Processing (ICIIP), 2011 International Conference on*, pages 1–6. IEEE, 2011.

[31] M. Mirza and S. Osindero. Conditional generative adversarial nets. *arXiv preprint arXiv:1411.1784*, 2014.

[32] H. Moudni, M. Er-rouidi, M. Oujaoura, and O. Bencharef. Recognition of amazigh characters using surf & gist descriptors. In *International Journal of Advanced Computer Science and Application. Special Issue on Selected Papers from Third international symposium on Automatic Amazigh processing*, pages 41–44, 2013.

[33] Y. Netzer, T. Wang, A. Coates, A. Bissacco, B. Wu, and A. Y. Ng. Reading digits in natural images with unsupervised feature learning. In *NIPS workshop on deep learning and unsupervised feature learning*, volume 2011, page 5, 2011.

[34] A. Odena, C. Olah, and J. Shlens. Conditional image synthesis with auxiliary classifier gans. *arXiv preprint arXiv:1610.09585*, 2016.

[35] A. Radford, L. Metz, and S. Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv preprint arXiv:1511.06434*, 2015.

[36] A. J. Ratner, C. M. De Sa, S. Wu, D. Selsam, and C. Ré. Data programming: Creating large training sets, quickly. In *Advances in Neural Information Processing Systems*, pages 3567–3575, 2016.

[37] A. J. Ratner, H. R. Ehrenberg, Z. Hussain, J. Dunnmon, and C. Ré. Learning to compose domain-specific transformations for data augmentation. *arXiv preprint arXiv:1709.01643*, 2017.

[38] S. Riedel, L. Yao, and A. McCallum. Modeling relations and their mentions without labeled text. *Machine learning and knowledge discovery in databases*, pages 148–163, 2010.

[39] L. Rothacker, S. Vajda, and G. A. Fink. Bag-of-features representations for offline handwriting recognition applied to arabic script. In *Frontiers in Handwriting Recognition (ICFHR), 2012 International Conference on*, pages 149–154. IEEE, 2012.

[40] R. E. Schapire and Y. Freund. *Boosting: Foundations and algorithms*. MIT press, 2012.

[41] C. Sun, A. Shrivastava, S. Singh, and A. Gupta. Revisiting unreasonable effectiveness of data in deep learning era. *arXiv preprint arXiv:1707.02968*, 2017.

[42] L. Theis, A. v. d. Oord, and M. Bethge. A note on the evaluation of generative models. *arXiv preprint arXiv:1511.01844*, 2015.

[43] P. Varma, B. He, P. Bajaj, I. Banerjee, N. Khandwala, D. L. Rubin, and C. Ré. Inferring generative model structure with static analysis. *arXiv preprint arXiv:1709.02477*, 2017.

[44] C. Vondrick, A. Khosla, T. Malisiewicz, and A. Torralba. HOGgles: Visualizing Object Detection Features. *ICCV*, 2013.

[45] H. Xiao, K. Rasul, and R. Vollgraf. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. *arXiv preprint arXiv:1708.07747*, 2017.

[46] Z. Yi, H. Zhang, P. T. Gong, et al. Dualgan: Unsupervised dual learning for image-to-image translation. *arXiv preprint arXiv:1704.02510*, 2017.

[47] D. Yoo, N. Kim, S. Park, A. S. Paek, and I. S. Kweon. Pixel-level domain transfer. In *European Conference on Computer Vision*, pages 517–532. Springer, 2016.

[48] K. Yu, Y. Lin, and J. Lafferty. Learning image representations from the pixel level via hierarchical sparse coding. In *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*, pages 1713–1720. IEEE, 2011.

[49] H. Zhang, T. Xu, H. Li, S. Zhang, X. Huang, X. Wang, and D. Metaxas. Stackgan: Text to photo-realistic image synthesis with stacked generative adversarial networks. *arXiv preprint arXiv:1612.03242*, 2016.

[50] M. Zhenjiang and Y. Baozong. Handwritten character recognition by extended loop neural networks. In *Speech, Image Processing and Neural Networks, 1994. Proceedings, ISSIPNN'94., 1994 International Symposium on*, pages 460–463. IEEE, 1994.

[51] J.-Y. Zhu, T. Park, P. Isola, and A. A. Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. *arXiv preprint arXiv:1703.10593*, 2017.