Supplementary Section

In this section, we include more details about our algorithm, labeling functions, datasets as well as additional results and comparisons, which could not be included in the main paper due to space constraints.

A. Algorithm

Algorithm 2 presents the overall stepwise routine of the proposed method, ADP , as described in Section 3. During the training phase, the algorithm updates weights of the model by estimating gradients for a batch of labeled data points. The hyperparameters that need to be provided include standard parameters that are provided while training a GAN, such as: (i) number of iterations of Algorithm 2; (ii) parameter k (similar to [18]) that describes how many times D and D_{LFB} would be updated with respect to G; and (iii) minibatch size m. Using empirical studies, we chose $m = 128$, $k = 2$ and number of iterations to be $60,000$.

B. Datasets

In this section, we provide more information on the datasets used for validating ADP in this work: MNIST, Fashion MNIST, SVHN and CIFAR 10. The MNIST dataset comprises 28×28 grayscale images (with one handwritten digit in each image) along with the corresponding label, with 50,000 training samples (image-label pairs). In case of SVHN, we used "format 2" of the dataset, which comprises 73257 32×32 images (each containing a digit captured from street views of house numbers) with the corresponding labels. In case of CIFAR 10, we merged five training batches of the dataset and built a training set of 50,000 images. This dataset contains RGB-images each of size of 32×32 spanning 10 classes: *automobile, airplane, bird, cat, deer, dog, frog, horse, ship, truck*. The total number of samples are almost equally distributed across all classes. Fashion MNIST, similar to MNIST, consists of a training set of 50,000 28×28 grayscale images with one of 10 classes: *Tshirt, Trouser, Pullover, Dress, Coat, Sandal, Shirt, Sneaker, Bag, Ankle boot*.

We also used the LookBook [47] dataset (Figure 10) to demonstrate cross-domain multi-task learning using ADP , as described in Section 5. This dataset contains 84,748 images across 17 classes: *Midi dress, mini dress, coat, jacket, fur jacket, padded jacket, hooded jacket, jumper, cardigan, knitwear, blouse, shirt, sleeveless tee, short sleeve tee, long sleeve tee, hoody, vest*. In this work, we grouped these 17 classes into 4 classes: *coat, pullover, t-shirt, dress*, in order to match with the Fashion MNIST dataset and thus help study cross-domain learning. We grouped coat, jacket, fur jacket, padded jacket, hooded jacket, jumper, cardigan to a single *coat* class; hoody to the *pullover* class; sleeveless tee, short sleeve tee to the *t-shirt* class; cardigan, knitwear, blouse, Midi dress, mini dress to the *Dress* class. Fashion MNIST dataset also has the same classes: *coat, pullover, t-shirt, Dress* among its label, thus facilitating our study.

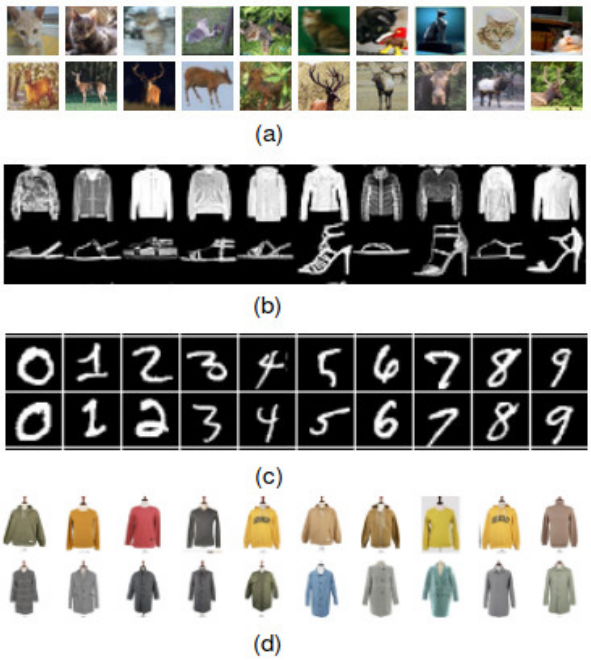

Figure 10: Sample images from the datasets studied in this work: (a) CIFAR 10, (b) Fashion MNIST, (c) MNIST, (d) LookBook

No additional pre-processing was performed on MNIST, Fashion MNIST, CIFAR 10 or the LookBook datasets. In case of SVHN, an additional crop was performed on each image to ensure only one digit is present in the image. The cropped image was subsequently sampled to maintain the 32×32 size. Figure 10 shows illustrative examples of images from the chosen datasets.

C. More on Labeling Functions

The Labeling Functions Block (LFB) in Figure 2a (Section 3) is implemented using the open-source framework, Snorkel [36]. We modified the underlying architecture of the Snorkel framework to include an adversarial approach, which otherwise estimates dependencies using MLE invoking Gibbs sampling. Labeling functions of three kinds: heuristics, image processing-based and deep learned features have been used in this work, as described in Section 4. Examples of labeling functions used in this work are shown as Labeling Functions 1, 2, 3 and 4. For each labeling function, a simple threshold rule on the L_2-norm of the aforementioned features is used. For each class of a dataset, the threshold information is obtained empirically as the average L_2-norm of the feature vectors of 20 random samples of that class (with α-trimming to remove outliers). It is worthy to mention that, for an abstract understanding of working process of our labeling functions, the return value of example

Algorithm 2: Training ADP

Input: Number of iterations, Number of steps to train D: k, Minibatch size: m

Output: Trained ADP model

for *number of iterations* **do**

  for k *steps* **do**

    Given noise prior $\mathbf{z} \sim \mathcal{N}(0, I)$, draw a batch of m samples from G: $\{(\tilde{\mathbf{x}}_1, \Theta_1, \Phi_1), \cdots, (\tilde{\mathbf{x}}_m, \Theta_m, \Phi_m)\}$;

    Use Equation 3 (from LFB) to compute probabilistic label vectors $\{\Lambda_1, \cdots, \Lambda_m\}$ given $\{(\tilde{\mathbf{x}}_1, \Theta_1, \Phi_1), \cdots, (\tilde{\mathbf{x}}_m, \Theta_m, \Phi_m)\}$;

    Draw a batch of m image-label pairs $((\mathbf{x}_1, y_1), \cdots, (\mathbf{x}_m, y_m))$ from real distribution $P_{real}(\mathbf{x}, y)$;

    Update weights of discriminators D and D_{LFB} (ψ_d and ψ_l respectively), using mini-batch stochastic gradient ascent with gradients as computed below:

$$\nabla_{\psi_d} \frac{1}{m} \sum_{i=1}^{m} \Big[\log D(\mathbf{x}_i, y_i) + \log(1 - D(\tilde{\mathbf{x}}_i, \Lambda_i)) \Big]$$

    and,

$$\nabla_{\psi_l} \frac{1}{m} \sum_{i=1}^{m} \Big[\log D_{LFB}(\Phi_{real_i}) + \log(1 - D_{LFB}(\Phi_i)) \Big]$$

  end

  Given noise prior $\mathbf{z} \sim \mathcal{N}(0, I)$, draw a batch of m samples from G: $\{(\tilde{\mathbf{x}}_1, \Theta_1, \Phi_1), \cdots, (\tilde{\mathbf{x}}_m, \Theta_m, \Phi_m)\}$;

  Update weights of generator G, ψ_g, using mini-batch stochastic gradient descent with gradients as computed below (each step below updated sequentially, one after another);

$$\nabla_{\psi_g} \frac{1}{m} \sum_{i=1}^{m} \Big[\log(1 - D_{LFB}(\Phi_i)) \Big]$$

  and

$$\nabla_{\psi_g} \frac{1}{m} \sum_{i=1}^{m} \Big[\log(1 - D(\tilde{\mathbf{x}}_i, \Lambda_i)) \Big]$$

end

Labeling Functions 3 and 4 are one-hot encoding. Though in practice we fit a nonlinear function to get a probabilistic output.

C.1. Ablation Studies with Labeling Functions

In order to understand the effect of different kinds of labeling functions, we performed an ablation study on the CIFAR10 dataset (considering it is the most natural of the considered datasets, and that it allows us to compute an Inception score to quantitatively compare the performance of various methods). In this study, we did not alter any hyperparameters described in Section 4. Our ablation study considers the following models:

M1: *ADP :* Full model

M2: *ADP with no dependencies:* Same model as ADP having 55 labeling functions, as in Table 3. Each labeling function is, however, considered *independent* of the other.

M3: *ADP with only heuristic labeling functions:* Same model as ADP with 36 heuristic labeling functions but without any image processing or deep learned feature-based labeling functions

M4: *ADP with only image processing labeling functions:* Same model as ADP with only 17 image processing-based labeling functions but without heuristic or deep learned feature-based labeling functions

M5: *ADP with only deep learned feature-based labeling functions:* Same model as ADP with only 2 deep learned feature-based labeling functions but without heuristic or image processing labeling functions

M6: *ADP with (heuristic labeling functions + deep learned feature-based labeling functions)*

M7: *ADP with (deep learned feature-based labeling functions + image processing labeling functions)*

Labeling Function 1: Sample heuristic labeling function (used for blobs in digits like: 0, 9, 6)

Input: Image
Output: Probabilistic label vector

```
/* Decision tree for English numerals
   recognition [24]                          */
```

if *blob(Image) == TRUE* **then**
  if *blob diameter(Image)* $\leq$ *0.5cm* **then**
    number = count blob(Image);
    if *number == 2* **then**
      return [0.2,0,0,0,0,0.1,0,0,0.6,0.1];
    end
    if *number == 1* **then**
      return [0.6, 0, 0.2, 0, 0.1, 0, 0, 0, 0.1]
    end
  end
  if *blob diameter(Image)* $>$ *0.5cm* **then**
    return [0.4, 0, 0, 0, 0.1, 0, 0.3, 0.1, 0.1]
  end
end

Labeling Function 2: Sample heuristic labeling function (used for digits with blob and stem like: 4, 6, 9)

Input: Image
Output: Probabilistic label vector

```
/* Decision tree for English numerals
   recognition [24]                          */
```

if *blob(Image) == TRUE* **then**
  number = count stem (Image);
  if *number == 0* **then**
    return [0.8, 0, 0, 0, 0.1, 0, 0, 0, 0.1]
  end
  if *number == 1* **then**
    return [0.1, 0, 0, 0, 0.4, 0, 0.4, 0, 0.1]
  end
  if *number == 2* **then**
    return [0, 0, 0, 0, 0.8, 0, 0, 0, 0.2]
  end
end

M8: *ADP with (image processing labeling functions + heuristic labeling functions)*

The inception scores for the aforementioned 8 models are presented in Table 7. The base ADP model comprising of all labeling functions outperforms all other models, highlighting the usefulness of a variety of labeling functions to model the non-trivial $P(\mathbf{x}, y)$ distribution.

D. More Qualitative Results

In addition to the results on CIFAR 10 presented in Section 4.4, we also studied the performance of our ADP method against other generative methods (CGAN, ACGAN, InfoGAN, CoGAN, TripleGAN) on MNIST, SVHN and Fashion MNIST datasets. Similar to CIFAR 10 generation, we changed the use case setup of the other methods to generate labeled images, using the publicly available code for each of the methods. Figures 11, 12 and 13 present these results.

Labeling Function 3: Sample image processing based labeling function (based on Bag-of-Words)

Input: Image, n: Number of classes
Output: Probabilistic label vector

```
/* NOTE: Loop presented below for purposes
   of clarity - it is implemented only once
   for a dataset                             */
```

for *i=1* $\cdots$ *n* **do**
  v_{avg_i} = average value of L_2 norm of Bag-of-feature() on subset of training samples from class i;
end
$\mathbf{v} = Bag - of - feature(Image)$;
$v_{image} = \|\mathbf{v}\|$;
return OneHot($\arg\min_i \left[|v_{avg_i} - v_{Image}|\right]$)

Labeling Function 4: Sample deep learned feature-based labeling function

Input: Image, n: Number of classes, Kernels from first layer of pre-trained AlexNet (trained on ImageNet)
Output: Probabilistic label vector

```
/* Deep learning based labeling function   */
```

m = Number of kernels from first layer of pre-trained AlexNet;
for *i=1* $\cdots$ *n* **do**
  for *j = 1* $\cdots$ *m* **do**
    $v_{avg_{ij}}$ = average value of Frobenius norm of activation map of j^{th} kernel on subset of training samples from class i;
  end
end
for *j = 1* $\cdots$ *m* **do**
  v_{Image_j} = value of Frobenius norm of activation map of j^{th} kernel on Image;
end
return OneHot($\arg\min_i[\mathbf{v}_{avg_i} \cdot \mathbf{v}_{Image}]$)

	M1	M2	M3	M4	M5	M6	M7	M8
Inception Score	**8.7**	4.32	5.52	4.91	4.73	7.01	7.52	7.27

Table 7: Ablation study w.r.t labeling functions, as described in Section C.1

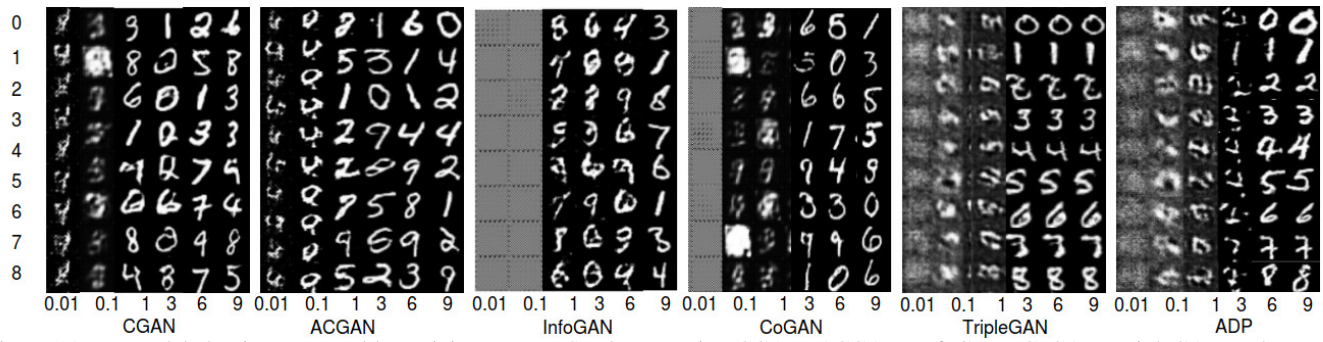

Figure 11: Image-label pairs generated by training on MNIST dataset using CGAN, ACGAN, InfoGAN, CoGAN, TripleGAN and our method, ADP . For a given model, the columns of images represents generations after 0.01k, 0.1k, 1k, 3k, 6k and 9k epochs, and the rows correspond to the associated class label. It is evident that from 6k epochs onward, ADP model starts generating quality images across classes and with a good image-to-label correspondence.

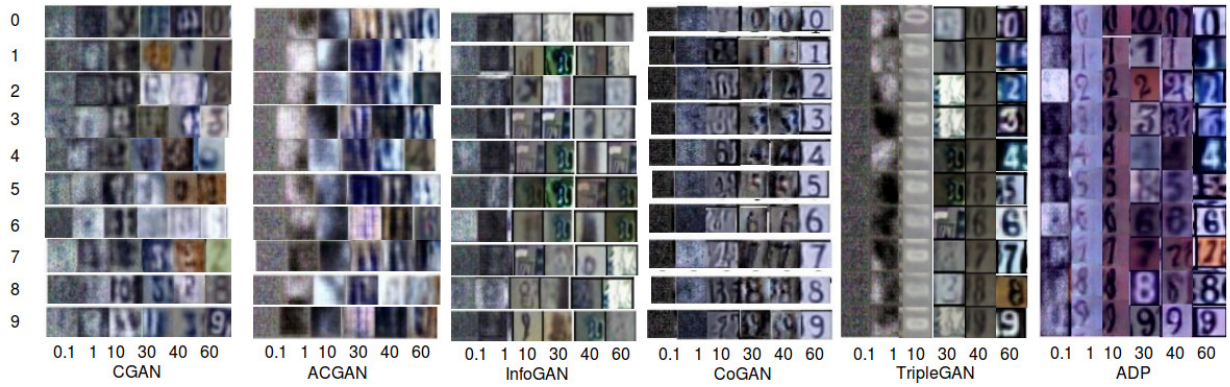

Figure 12: Image-label pairs generated by training on SVHN dataset using CGAN, ACGAN, InfoGAN, CoGAN, TripleGAN and ADP . For a given model, the columns of images represents generations after 0.1k, 1k, 10k, 30k, 40k and 60k epochs, and the rows correspond to the associated class label.

MNIST: Figure 11 shows that both our method ADP and TripleGAN generate good quality images on the MNIST dataset. We observe that both ADP and TripleGAN give a high image-to-label correspondence. Surprisingly, state-of-the-art methods such as CGAN, ACGAN, InfoGAN and CoGAN fail to capture image-to-label correspondence despite generating good quality images.

SVHN: As shown in Figure 12, we observe that our method generates human-recognizable images with a good image-to-label correspondence in just 1k epochs on the relatively harder SVHN dataset. At higher epochs, CoGAN (epoch = 30) and TripleGAN (epoch = 40) also generate images of good quality, but broadly fail to capture different styles, backgrounds and illuminations of the generated digit.

Fashion MNIST: Most of the considered methods do well on this dataset. ADP and TripleGAN provide the sharpest results on close visual observation (Please see Figure 13).

E. More Quantitative Results

Parzen Window Based Evaluation: In addition to the results with Inception score and *HTT* presented in Section 4.4, we compared our method against other generative models (described in Section 4) based on the Parzen window score at test time. Parzen window [5] is a commonly used non-parametric density estimation method to evaluate generative models (especially GANs [18]) for which exact likelihood is not tractable. Based on the samples generated by the model, we use a Parzen window with a Gaussian kernel as a density estimator. This helps obtain a proxy for true log-likelihood and thereby evaluate test log-likelihood. These results are shown in Table 8. The table shows that ADP has performed significantly well on MNIST (Score is 344) and SVHN (Score is 246) dataset and outperformed other state-of-the-art models including TripleGAN. For Fashion MNIST, our method is a close second with respect to TripleGAN. We chose the Parzen window size using cross-validation, as described in [18].

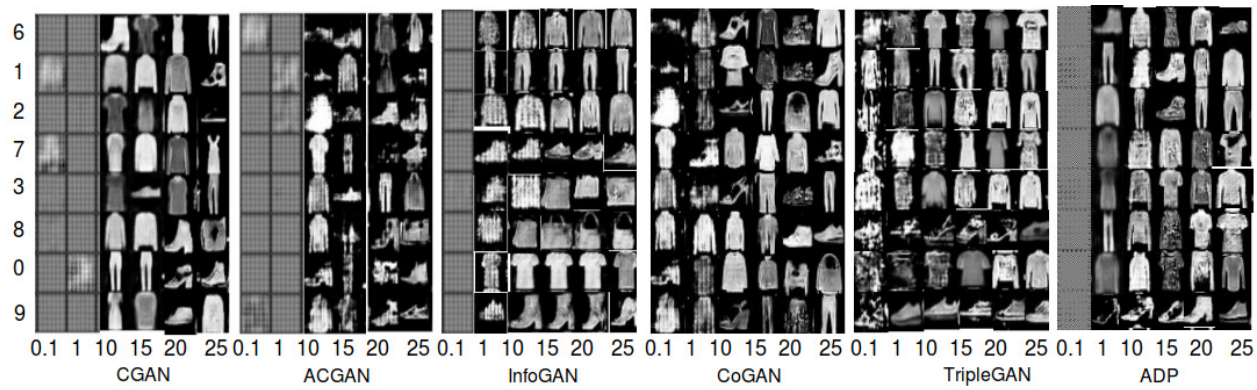

Figure 13: Image-label pairs generated by training on Fashion MNIST dataset using CGAN, ACGAN, InfoGAN, CoGAN, TripleGAN and our method, ADP . For a given model, the columns of images represents generations after 0.1k, 1k, 10k, 15k, 20k and 25k epochs, and the rows correspond to the associated class label.

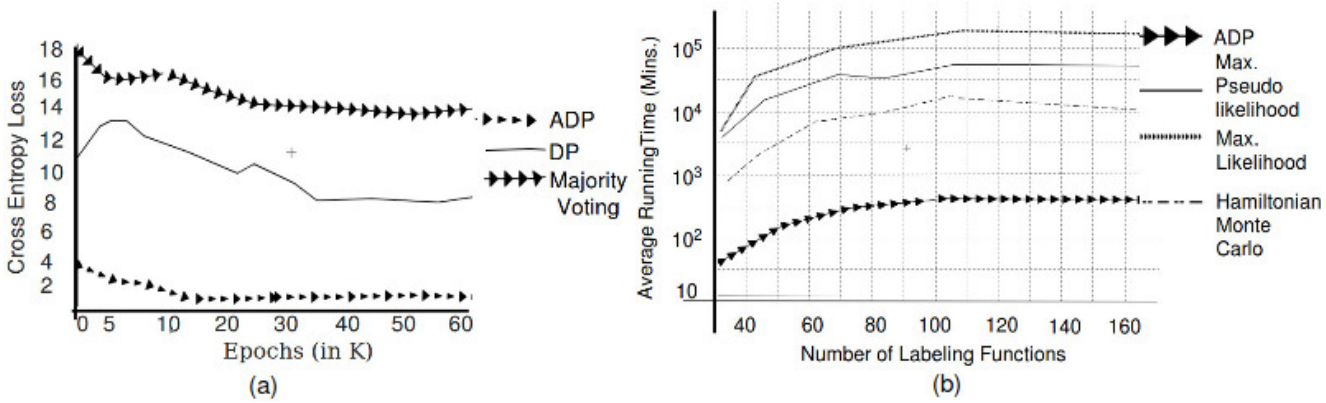

Figure 14: (a) Test-time classification cross-entropy loss of a pre-trained ResNet model on image-label pairs generated by ADP, ADP (i.e. only its Image-GAN component) with majority voting and ADP (i.e. only its Image-GAN component) with DP for labels; (b) Average running time of ADP against other methods to estimate the relative accuracies and inter-function dependencies in DP.

F. More Results on Multi-task Joint Distribution Learning

In continuation to the results presented in Section 5 (and 1), we present more results for the capability of ADP to perform multi-task joint distribution learning in Figure 16. The figure captures our promise and shows that ADP is able to generate samples from two different domains, including samples of different colors.

G. Comparison against Vote Aggregation Methods

Comparison against Majority Voting and DP: To study the usefulness of learning relative accuracies and inter-function dependencies using ADP , we compared the performance of our method, both with majority voting and Data Programming (DP [36]). In majority voting, LFB does not estimate relative accuracies and inter-function dependencies of labeling functions as described in Section 3. Instead, for a given image, each labeling function of LFB makes a probabilistic prediction, and we take a maximum vote to obtain the final label. As in Section 4.4, we studied the test-time classification cross-entropy loss of a pre-trained ResNet model on image-label pairs generated by ADP, ADP (i.e. only its Image-GAN component) with majority voting and DP. The results are presented in Figure 14a, which shows that ADP has significantly lower cross-entropy loss than the other two methods, thus corroborating its effectiveness.

Adversarial Data Programming vs MLE-based Data Programming: To further quantify the benefits of our ADP , we also show how our method compares against Data Programming (DP) [36] using different variants of MLE: MLE, Maximum Pseudo-likelihood, and Hamiltonian Monte Carlo. We note that DP only aggregates labels; we hence, combined a vanilla GAN with DP as separate components to conduct this study. We started with a small number of labeling functions (viz., 35 functions) and progressively added additional labeling functions, noting the

	GAN	CGAN	ACGAN	InfoGAN	CoGAN	ADP	Triple
MNIST	198	201	204	225	278	**344**	321
FMNIST	213	206	234	276	254	292	**312**
SVHN	87	145	178	158	123	**246**	223

Table 8: Parzen window based evaluation on MNIST, FMNIST and SVHN datasets.

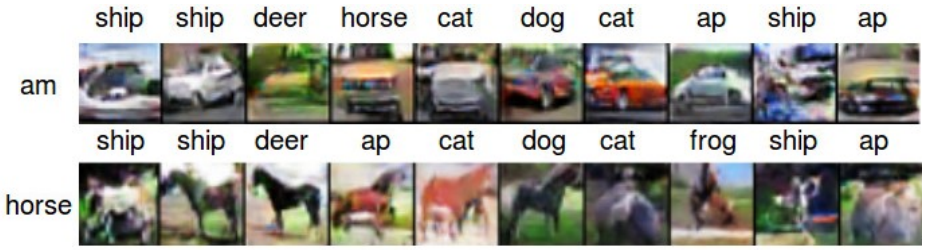

Figure 15: Sample results of image-label pairs generated by combining a vanilla GAN (for image generation) and DP [36] (for label generation) using the same labeling functions used in this work. Row labels represent the original class label (am = automobile) and column labels are provided by DP. Note the poor image-label correspondence, supporting the need for our work.

Figure 16: Demonstration of cross-domain multi-task learning using ADP : (a)(b) Generated samples of Shirt (class 6 of Fashion MNIST dataset); (c) Generated samples of T-shirt (class 0 of Fashion MNIST dataset). Samples generated of the LookBook dataset are color images (top rows), while those of Fashion MNIST are grayscale images (bottom rows).

time taken by each aforementioned parameter estimation method. Figure 14b presents the results and shows that ADP is almost 100X faster than MLE-based estimation. Figure 15 also shows sample images generated by the vanilla GAN, along with the corresponding label assigned by MLE-based DP using the same labeling functions as used in our work. Clearly, the labels are incorrect, thus supporting the value of the proposed work in learning a joint distribution, than combining two individual components.