Dense Crowd Counting Convolutional Neural Networks with Minimal Data using Semi-Supervised Dual-Goal Generative Adversarial Networks

Greg Olmschenk[1] Jin Chen[2] Hao Tang[3] Zhigang Zhu[1,2]
[1]The Graduate Center of the City University of New York
[2]The City College of New York - CUNY
[3]Borough of Manhattan Community College - CUNY
golmschenk@gradcenter.cuny.edu, jchen025@citymail.cuny.edu,
htang@bmcc.cuny.edu, zhu@cs.ccny.cuny.edu

Abstract

In this work, we generalize semi-supervised generative adversarial networks (GANs) from classification problems to regression for use in dense crowd counting. In the last several years, the importance of improving the training of neural networks using semi-supervised training has been thoroughly demonstrated for classification problems. This work presents a dual-goal GAN which seeks both to provide the number of individuals in a densely crowded scene and distinguish between real and generated images. This method allows the dual-goal GAN to benefit from unlabeled data in the training process, improving the predictive capabilities of the discriminating network compared to the fully-supervised version of the network. Typical semi-supervised GANs are unable to function in the regression regime due to biases introduced when using a single prediction goal. Using the proposed approach, the amount of data which needs to be annotated for dense crowd counting can be significantly reduced.

1. Introduction

Every year, gatherings of thousands to millions occur for protests, festivals, pilgrimages, marathons, concerts, and sports events. For any of these events, there are countless reasons to desire to know how many people are present. For those hosting the event, both real-time management and future event planning is dependent on how many people are present, where they are located, and when they are present. For security purposes, knowing how quickly evacuations can be executed and where crowding might pose a threat to individuals is dependent on the size of the crowds. In journalism, crowd sizes are frequently used to measure the significance of an event, and systems which can accurately report on the event size are important for a rigorous evaluation.

While GANs have already shown significant potential in semi-supervised training, they have only been used for a limited number of cases. In particular, they have almost exclusively been used for classification problems thus far. In this work, we propose a dual-goal semi-supervised GAN for the regression problem of dense crowd counting. Although the switch from classification to regression may initially seem to be a trivial extension, the nature of a GAN's optimization function results in a non-trivial obstacle. The discriminating portion of the GAN must have the objective of labeling the fake data from generating portion as fake. In a classification semi-supervised GAN, an additional "fake" class is added to the possible list of classes. However, in regression, where the data is labeled with real-valued numbers, deciding what constitutes a "fake" labeling is not straight forward. To avoid this problem, we use a dual-goal GAN (DG-GAN) approach, where the network outputs two labels: the desired regression prediction and a real/fake classification.

This work presents three primary contributions:

1. A dual-goal GAN architecture allowing for semi-supervised training for dense crowd counting.

2. An analysis demonstrating the improved accuracy when using limited labeled data of the DG-GAN architecture compared to an identical CNN architecture without the generator component.

3. A series of experiments comparing the accuracy improvements gained by the DG-GAN using various quantities of labeled and unlabeled examples.

The remainder of this work is structured as follows. First, we provide related work on semi-supervised GANs and dense crowd counting in Section 2. Next, we detail the optimization goals and architecture of the DG-GAN and

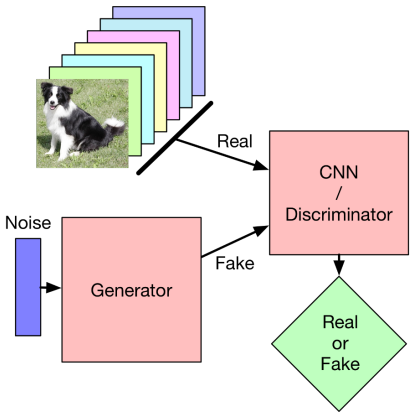

Figure 1: The structure of a basic GAN. Real and fake images are fed to a discriminator network, which tries to determine whether the images are real or fake. The fake images are produced by a generator network.

explain the experimental setup in Section 3. Afterward, the results of our experiments and an analysis of them are provided in Section 4. Finally, we provide some closing remarks in Section 5.

2. Background and Related Work

A Generative Adversarial Network (GAN) [3] consists of two neural networks which compete against one another. One of the networks generates fake data; hence it is typically called the generator. The other network attempts to distinguish between real data and the fake generated data; consequently, this network is called the discriminator. Both networks are trained together, each continually working to outperform the other and adapting in accordance to the other.

Though GANs are now fairly common, to provide s groundwork understanding for our DG-GAN, it is worth revisiting the details of a GAN from the viewpoint of probability distributions. Although the GAN approaches can be used for any prediction application, to give a concrete understanding, these explanations are given in terms of computer vision problems, specifically where the datasets consist of individual images. This means an example of real data (and thus the input of the discriminator) is an image, and the output of the generator is likewise an image. The overarching structure of a GAN can be seen in Figure 1.

The generator network takes random noise as input (usually a vector with each value sampled from a normal distribution) and outputs a generated image. The discriminator takes as input images and outputs a binary classification of either fake or real data. Images can be represented by a vector, with each element representing the value of a pixel in the image[1]. In any image, each element of this vector has a value within a certain range representing the intensity of that pixel. For this explanation, we will state the minimum element value (pixel value) as being 0, and the maximum as being 1. Of course, this vector can be represented as a point in N dimensional space, where N is the number of elements in the vector. The possible positions of an image's point are restricted to the N dimensional hypercube with a side length of 1. Here, it is important to note that real-world images are not equally spread throughout this cube. That is, most points in the cube correspond to images that would look like random noise to a human. Images from the real world usually have properties like local consistency in both texture and color, the logical relative positioning of shapes, etc. Real world images lie on a manifold within the cube [2]. Subsets of real-world images, such as the set of all images containing a dog, lie on yet a smaller manifold. This manifold represents a probability distribution of the real world images. We can view the real world as a data generating probability distribution, with each position on the manifold having a certain probability based on how likely that image is to exist in the real world.

The goal of the generator is then to produce images which match the probability distribution of the manifold as closely as possible. Input to the generator is a point sampled from the probability distribution of (multidimensional) random normal noise, and the output is a point in the hypercube–an image. The generator is then a function which transforms a normal distribution into an image data distribution. Formally,

$$p_{fake}(\boldsymbol{x}) = G(\mathcal{N}) \tag{1}$$

where G represents the generator function, $\boldsymbol{x}$ is a random variable representing an image, $\mathcal{N}$ is the normal distribution, and $p_{fake}(\boldsymbol{x})$ is the probability distribution of the images generated by the generator. The desired goal of the generator is to minimize the difference between the generated distribution and the true data distribution. One of the most common metrics to minimize this difference is the Kullback-Leibler (KL) divergence between the generator distribution and the true data distribution using maximum likelihood estimation. This is done by finding the parameters of the generator, $\boldsymbol{\theta}$, which produce the smallest divergence,

$$\boldsymbol{\theta}^* = \arg\min_{\boldsymbol{\theta}} D_{\mathrm{KL}}(p_{real}(\boldsymbol{x}) \parallel p_{fake}(\boldsymbol{x};\boldsymbol{\theta})). \tag{2}$$

To find this set of parameters, each of the discriminator and the generator works toward minimizing a loss function. For the discriminator, the loss function is given by

$$\begin{aligned} L_D = &-\mathbb{E}_{\boldsymbol{x}\sim p_{real}(\boldsymbol{x})}[\log D(\boldsymbol{x})] \\ &-\mathbb{E}_{\boldsymbol{x}\sim p_{fake}(\boldsymbol{x})}[\log(1 - D(\boldsymbol{x}))] \end{aligned} \tag{3}$$

and the generator's loss function is given by

$$L_{fake} = -\mathbb{E}_{\boldsymbol{x}\sim p_{fake}(\boldsymbol{x})}[\log(D(\boldsymbol{x}))]. \tag{4}$$

[1]One element per pixel is in the case of grayscale images. For RGB images, there will be three elements in the vector for each pixel, one for each color channel of the pixel.

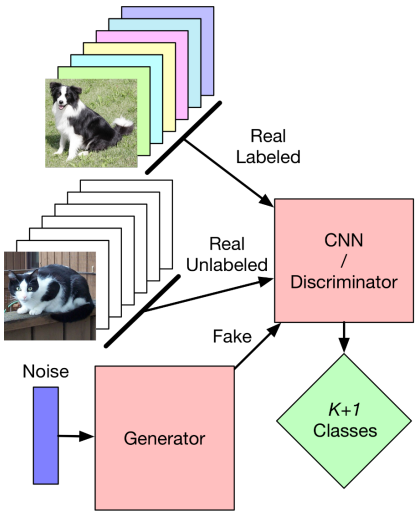

Figure 2: The structure of a semi-supervised GAN. Both labeled and unlabeled real images, as well as fake images, are fed to a discriminator network, which tries to determine which class each image belongs to (K real classes and one fake class). The discriminator wishes to label images from the generator as belonging to a special "fake" class.

In the case of image data, this approach has led to generative models which can produce realistic looking images reliably [7].

For a semi-supervised classification GAN, both a labeled and an unlabeled dataset is used, and in addition to distinguishing between real and fake, the discriminator also tries to label a real input data sample into one of the given classes. The primary goal of this type of GAN is to allow the discriminator's prediction task to be trained with relatively small amounts of labeled data using unlabeled data to provide the network with additional information. As unlabeled data is usually much easier to obtain than labeled data, this provides a powerful means to reduce the requirements of training neural networks. This semi-supervised GAN structure can be seen in Figure 2.

Where in a simple GAN the discriminator would be passed true examples and fake examples, in the semi-supervised GAN the discriminator is given true labeled examples, true unlabeled examples, and fake examples. We can better understand why this is useful by considering the case of image classification. In this case, the discriminator is being trained to predict the correct class of a true image, which can be one of the K classes that exist in the dataset. The discriminator is given the additional goal of attempting to label any fake images with a $K+1$th class, which only exists to label fake data (i.e., does not exist in the true label dataset). For the case of unlabeled, all we know is that it must belong to one of the first K classes, as the $K+1$th class does not exist in the real data. The discriminator is then punished for labeling true unlabeled data as the $K+1$th class. This is useful because the discriminator cannot simply overfit to the labeled data, as it still has to accommodate for the unlabeled data. At the same time, the fake data prevents the discriminator from allowing simple features to be the deciding factor, as the generator is able to produce such simple features.

To understand what is happening in this semi-supervised learning more intuitively, we can imagine the extreme case of an ideal discriminator and generator. The generator would have to have learned to produce data which exactly matches the true data distribution. For this to happen, the discriminator must have forced the generator to learn this (as the generator's training is entirely dictated by backpropagation from the discriminator), meaning the discriminator too "knows" exactly the data distribution. If there were any difference between the true and generated image distributions, the discriminator could use this to distinguish between real and fake, and then the generator could still be trained further toward producing a match of the true distribution.

Viewing this from the perspective of the manifold in data space again, there are few labeled data points and many unlabeled data points which must lie on the manifold. The manifold has different regions (or even separate manifolds) for each class, but even the unlabeled data has to lie somewhere on the manifold. As the discriminator trains, it learns how to segment the data points into categories. To do this, it creates a mapping from a predictive manifold to a class, with the training warping the manifold to contain each of the data points for that class. At the same time, the generator prevents the manifold from warping too severely to reach data points in arbitrary ways. Intuitively, this is because severely warping the manifold to reach true data points can result in the manifold stretching into the area which does not represent true images. The generator acts a pressure on the manifold to reduce this. By generating images near the manifold, the generator forces the discriminator's manifold not to wander into areas that don't contain real images. In this sense, the generator is a form of regularization for the discriminator, but one which is based on real-world data.

As originally formulated by [8], the discriminator loss function is then defined by

$$L_D = L_{supervised} + L_{unsupervised} \tag{5}$$

$$L_{supervised} = - \mathbb{E}_{\boldsymbol{x},y \sim p_{labeled}(\boldsymbol{x},y)} \log[p_{model}(y \mid \boldsymbol{x}, y < K+1)] \tag{6}$$

$$L_{unsupervised} = - \mathbb{E}_{\boldsymbol{x} \sim p_{unlabeled}(\boldsymbol{x})} \log[1 - p_{model}(y = K+1 \mid \boldsymbol{x})] - \mathbb{E}_{\boldsymbol{x} \sim p_{fake}} \log[p_{model}(y = K+1 \mid \boldsymbol{x})]. \tag{7}$$

As for the generator, the first option for a loss function is the straight forward one which aims to have the discriminator

label the fake images as from real classes. Specifically,

$$L_G = -\mathbb{E}_{\boldsymbol{x} \sim p_{fake}} \log[p_{model}(y < K+1 \mid \boldsymbol{x})]. \quad (8)$$

Since their development, semi-supervised GANs have been used to improve training in many areas of classification, including digit classification [12, 13, 8], object classification [12, 13, 8], facial attribute identification [13], and image segmentation (per pixel object classification) [11].

3. Methodology

3.1. Semi-supervised Dual-Goal Regression GAN

While semi-supervised GANs have been highly successful in training to reduce data requirements for classification problems, the approach does not directly translate to regression problems, such as dense crowd counting. This is due to the requirement of a fake class being added to the possible classifications. In regression problems, the label spans real numbers rather than discrete classes. Furthermore, choosing a specific number or range of numbers to represent a fake label introduces bias to the system. For example, in the case of crowd counting, negative numbers could be chosen to represent a fake label. However, we can imagine an extreme case where the generator and discriminator train to perfect. If the generator is able to produce images which exactly match those from the real distribution, then the best value for the discriminator to pick would be half value of the number of people in the image. This is because if the image was real, the correct count should be picked. If the image was fake, a negative number should be selected. Since we are assuming the generator can produce images which match the real distribution, the minimal discriminator loss will come from predicting a middle value. Even though the generator will never train to this level of perfection, this effect results in a significant overall bias nonetheless.

Our solution to the above issue is a semi-supervised dual-goal GAN (DG-GAN). This GAN requires the discriminator to provide two separate outputs: the desired regression value and a classification of whether the input example is part of the real or generated distribution. The approach does not present a bias in the regression value prediction but forces the network to share parameters for the prediction of whether or not the example is real.

Formally, the objective of the discriminator in our DG-GAN is then given by,

$$L_D = L_{supervised} + L_{unsupervised} \quad (9)$$

$$L_{supervised} = \mathbb{E}_{\boldsymbol{x}, y_s \sim p_{data}(\boldsymbol{x}, y)}[(D(\boldsymbol{x}) - y_s)^2] \quad (10)$$

$$\begin{aligned} L_{unsupervised} = & \\ & -\mathbb{E}_{\boldsymbol{x} \sim p_{unlabeled}(\boldsymbol{x})} \log[p_{model}(y_u = 0 \mid \boldsymbol{x})] \\ & -\mathbb{E}_{\boldsymbol{x} \sim p_{fake}} \log[p_{model}(y_u = 1 \mid \boldsymbol{x})], \end{aligned} \quad (11)$$

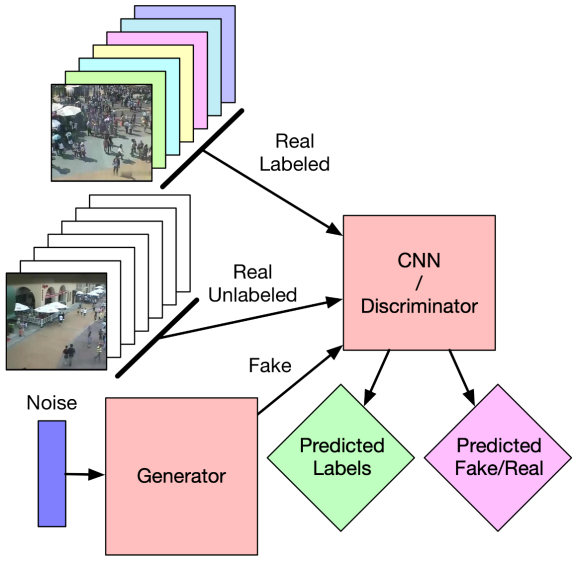

Figure 3: The structure of a semi-supervised dual-goal GAN. A regression label is predicted alongside a binary real/fake classification label. This structure allows for improved accuracy through semi-supervised training while preventing the bias that can be introduced by more naive methods.

where y_s is the supervised regression label, and y_u is the unsupervised real/fake binary classification label.

The generator is trained to optimize the likelihood that the generated images are considered real by the discriminator.

$$L_G = -\mathbb{E}_{\boldsymbol{x} \sim p_{fake}} \log[p_{model}(y_u = 0 \mid \boldsymbol{x})]. \quad (12)$$

This combination of supervised regression and unsupervised classification forces the discriminator to learn more robust features of crowd images, enabling it to perform well even with limited labeled training data. Equation (10) provides a standard mean squared error loss function allowing the discriminator to improve the regression prediction, and no other loss *directly* competes for to alter this value. Instead, the unsupervised loss acts as a form of regularization based on the real distribution of images. This forces the discriminator to function well even on the unlabeled images, and learn to recognize more robust patterns to avoid generated images. This overall model structure is shown in Figure 3.

The generator portion of the network and the real/fake binary classification goal is only used when training the DG-GAN. During test evaluation, only the discriminator portion of the DG-GAN is used. In our experiments, the discriminator of the DG-GAN is identical to the CNN we compare against. This helps to focus the results on the benefits of the DG-GAN training method as opposed to the CNN/discriminator network architecture details.

3.2. Experimental Setup

For our experiments, we used the UCF-QNRF dataset [6]. This dataset contains 1535 total images split as 1201 training images and 334 testing images. These training and testing distributions are provided by the dataset provider. The dataset contains 1,251,642 total head count, with a median of 425 (per image), and a mean of 815.4. To the best of our knowledge, the UCF-QNRF dataset is currently the largest dataset in terms of the number of head counts. With a minimum of 49 and a maximum of 12,865, the dataset contains an enormous variety of crowd density levels. Furthermore, the camera perspective angles range drastically from near parallel to ground level to nearly perpendicular with the ground. Lighting conditions, environmental scenes, pixel size of individuals, and levels of occlusion are similarly widely varied. Typical example images from the dataset are shown in Figure 4. The UCF-QNRF dataset provides a challenging and extensive dataset for dense crowd counting.

Images in the UCF-QNRF vary widely in resolutions. Our network accepts image patches of 224×224. The network is trained to predict the number of head counts within random training image patches. During the test phase, an overlapping sliding window is used to make a prediction of the number head counts in each patch of the image. Overlapping values are averaged, and the final prediction for an image under is given by the sum of these values.

[6] showed that a standard DenseNet [4] architecture was able to outperform many application-specific networks [9, 10, 1, 15] for crowd counting. Though [6] also provides an extended application specific version of DenseNet, we chose to use the original version of DenseNet201 as the discriminator in our experiments. This simplifies the repeatability of the experiment while still providing a network comparable to the state-of-the-art in terms of accuracy. This network structure is shown in Table 1 For the generator, we use the generator architecture from the well-known DCGAN [7]. This generator has been shown to function well on a wide variety of applications. The details of our generator architecture are shown in Table 2.

For each experiment, we use the discriminating CNN network with and without the generator. Of course, the binary classification goal of real/fake is only included in the case which includes the generator. This compares the standard CNN network against the identical network, with the only changes being the inclusion of the generator feeding data to the CNN and the real/fake prediction goal. This allows us to isolate the value of the DG-GAN compared to the CNN without placing undue importance on the details of the CNN/discriminator architecture.

Experiments with a greater number of labeled training images include all images used by experiments with a smaller number of labeled training images. That is, the experiment using 20 labeled images includes all labeled images in the experiment using 10 labeled images. This ensures that increased accuracy in one experiment is the result of increased

Layers	Output Size	Filter Types
Convolution	112 × 112	7 × 7 conv, stride 2
Pooling	56 × 56	3 × 3 max pool, stride 2
Dense Block (1)	56 × 56	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 6$
Transition Layer (1)	56 × 56	1 × 1 conv
	28 × 28	2 × 2 average pool, stride 2
Dense Block (2)	28 × 28	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 12$
Transition Layer (2)	28 × 28	1 × 1 conv
	14 × 14	2 × 2 average pool, stride 2
Dense Block (3)	14 × 14	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 48$
Transition Layer (3)	14 × 14	1 × 1 conv
	7 × 7	2 × 2 average pool, stride 2
Dense Block (4)	7 × 7	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 32$
Regression Layer	1 × 1	7 × 7 average pool

Table 1: The discriminator architecture based on DenseNet201. The growth rate for the filters is $k = 32$ (which defines the number of channels per DenseNet201 specifications). Each convolution is followed by a leaky ReLU, except the final convolution.

Layers	Output Size	Output Filters
Transposed Convolution (1)	14 × 14	512 features
Transposed Convolution (2)	28 × 28	256 features
Transposed Convolution (3)	56 × 56	128 features
Transposed Convolution (4)	112 × 112	64 features
Transposed Convolution (5)	224 × 224	3 features

Table 2: The generator architecture based on DCGAN. Each layer is followed by a leaky ReLU, except the final layer which is followed by a Tanh.

Figure 4: Typical examples of crowd scene images from the UCF-QNRF dataset with various levels of densities, illuminations, and distributions.

available labeled data rather than due to the random selection of training data. The CNN and DG-GAN are trained with the same labeled images in all cases. The data selected for training is randomly sampled from the entire UCF-QNRF training dataset. This random sampling is seeded for repeatability.

All code and hyperparameters are given at https://github.com/golmschenk/sr-gan.

4. Results

For each of our experiments, we provide the mean absolute error (MAE), normalized absolute error (NAE), and root mean squared error (RMSE). These are given by the following equations:

$$\text{MAE} = \frac{1}{N}\sum_{i=1}^{N}\left|\hat{C}_i - C_i\right| \tag{13}$$

$$\text{NAE} = \frac{1}{N}\sum_{i=1}^{N}\frac{\left|\hat{C}_i - C_i\right|}{C_i} \tag{14}$$

$$\text{RMSE} = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(\hat{C}_i - C_i)^2} \tag{15}$$

MAE and RMSE tend to be the standard metrics of interest within the dense crowd counting community. [6] recently introduced NAE to capture the degree of miscount relative to the person count for each image. This tends to emphasize images with smaller counts more than the other two measures. We note that the supervised loss of the network uses the RMSE. This is a commonly used supervised loss function. However, it does not directly train to optimize NAE, as such, this measure tends to be less consistent. For all experiments, the evaluations given are for the test dataset as designated by the dataset provider.

In all experiments, the network architecture of the CNN and the discriminator are identical. Notably, these are the only portions of the network used during evaluation of the test data (the generator is only used during training). This emphasizes the value gained by the DG-GAN training method rather than the architecture of the inference network.

The first set of experiments uses a limited set of labeled training images. The unlabeled training images consist of any available training image not included in the labeled set, resulting in a number of unlabeled images being 1201 minus the number of labeled images. The results from this set of experiments are shown in Table 3.

We see from this set of experiments that the DG-GAN significantly outperforms the CNN in every case for MAE and RMSE. NAE results are less consistent. For MAE, the DG-GAN often outperforms the CNN even when it uses two or four times as much data. For example, the DG-GAN using 80 labeled examples significantly outperforms the CNN using 320 labeled examples. In some cases, the difference is less significant, such as when 40 examples were used. However, the model with the simple addition of the generator and the losses to the CNN always outperforms the ordinary CNN. Similarly, the RMSE of the DG-GAN always outperforms the CNN using the same amount of data. These results mirror the MAE case, with the DG-GAN with 80 labeled examples once again outperforming the CNN using 320 labeled examples.

The second set of experiments consists of using both a limited number of labeled examples and a limited number of unlabeled examples. In particular, each experiment uses ten

	MAE		RMSE		NAE	
Labeled examples	CNN	DG-GAN	CNN	DG-GAN	CNN	DG-GAN
10	422.9	303.0	705.2	504.0	0.5369	0.5136
20	390.1	293.4	643.6	474.5	0.7076	0.4917
40	285.0	259.8	487.2	391.1	0.4823	0.5452
80	260.2	197.8	432.4	305.2	0.4853	0.3512
160	237.4	194.8	385.4	295.2	0.4735	0.4003
320	227.6	186.9	362.4	298.7	0.4228	0.3237

Table 3: Results using varying levels of labeled training examples and using all remaining training examples as unlabeled data. In each experiment, the CNN and the discriminator network architectures are identical.

Examples		MAE		RMSE		NAE	
Labeled	Unlabeled	CNN	DG-GAN	CNN	DG-GAN	CNN	DG-GAN
10	100	422.9	332.1	705.2	531.3	0.5369	0.5694
20	200	390.1	295.7	643.6	461.3	0.7076	0.6082
30	300	323.3	279.2	525.2	416.0	0.5878	0.6394
40	400	285.0	259.4	487.2	416.3	0.4823	0.4664
50	500	277.4	230.9	489.2	372.8	0.4881	0.4566

Table 4: Results using varying levels of labeled training examples and ten times as many unlabeled examples. In each experiment, the CNN and the discriminator network architectures are identical.

times as many unlabeled examples as labeled examples. The results from this set of experiments are shown in Table 4. The readers may compare Table 3 and Table 4 for the performance of under the same numbers of labeled examples (e.g., 10, 20, and 40) but with different numbers of unlabeled examples (e.g., 1201-10 compared to 100, 1201-20 compared to 200, 1201-40 compared to 400). It should be noted, the greatest number of labeled examples used in this set of experiments is far smaller than in the previous set of experiments. This is due to the limited total dataset size (large numbers of labeled examples would not have sufficient proportional amounts of unlabeled examples to train with).

Once again, this set of experiments clearly show the advantage of adding the generator and DG-GAN to the original CNN. In every case, the MAE and RMSE of the DG-GAN significantly outperform the CNN, often even versions of the CNN with more data. For example, the DG-GAN with 20 examples achieves an MAE which outperforms the CNN with 30, 40, and 50 labeled examples.

4.1. Discussion

Using the DG-GAN, we can significantly increase the performance of the CNN. The only changes made to the network structure are the addition of the generator and the additional output to provide the real/fake binary classification loss. Although this increases the training time significantly, the benefits of increased accuracy justify additional training.

The second set of experiments verifies that the benefit of the DG-GAN does not require enormous unlabeled datasets relative to the size of the labeled dataset, while the first set of experiments shows what can be obtained using all available unlabeled images for varying sizes of labeled images.

The second set of experiments shows that even with a much lower number of unlabeled examples the DG-GAN still provides a significant benefit. For 10 labeled examples and 100 unlabeled examples, the DG-GAN has resulted in a significant accuracy increase. However, additional insight can be gleaned by comparing the two tables of experiments. The DG-GAN using 10 labeled examples and 1191 unlabeled examples provides a more significant benefit to the DG-GAN. This demonstrates that the DG-GAN can benefit from large amounts of unlabeled data. Notably, in a real use case, unlabeled data is extremely easy to obtain (more pictures simply need to be taken), while labeled data is extremely costly to produce.

5. Conclusions and Future Work

This work provided the DG-GAN model, a means by which semi-supervised dense crowd counting networks can be trained. We've shown the method significantly improves the performance of a CNN with limited data training with

the UCF-QNRF dataset. Equivalently, the DG-GAN can train to the same level of accuracy as its counterpart CNN using far less labeled data.

While this already convincingly demonstrates the effectiveness of the DG-GAN, there are several ways we intend to expand the outcomes of this work. Most notably, similar experiments will be performed on additional well-known datasets such as the ShanghaiTech dataset [15], the UCF-CC-50 dataset [5], and the World Expo dataset [14]. However, these datasets have a more limited number of training images, restricting how varied the dataset sizes can be. Furthermore, the UCF-QNRF dataset provides one of the most diverse sets of images available. The remaining extension of this work is to provide a more complete analysis of the number of examples required for both labeled and unlabeled datasets to achieve various comparable levels of accuracy.

While these additional experiments will more thoroughly define the capabilities and limitations of the DG-GAN, the results in Section 4 already establish the unmistakable potential for the DG-GAN for use in dense crowd counting.

6. Acknowledgments

This research is supported by the National Science Foundation via awards #CNS-1737533 and #IIP-1827505, and by Bentley Systems, Inc., through a CUNY-Bentley Collaborative Research Agreement (CRA). Additional travel support provided by the Defense Intelligence Agency via the Rutgers University Consortium for Critical Technology Studies.

References

[1] V. Badrinarayanan, A. Kendall, and R. C. SegNet. A deep convolutional encoder-decoder architecture for image segmentation. *arXiv preprint arXiv:1511.00561*. 5

[2] C. Fefferman, S. Mitter, and H. Narayanan. Testing the manifold hypothesis. *Journal of the American Mathematical Society*, 29(4):983–1049, 2016. 2

[3] 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. 2

[4] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger. Densely connected convolutional networks. In *CVPR*, volume 1, page 3, 2017. 5

[5] H. Idrees, I. Saleemi, C. Seibert, and M. Shah. Multi-source multi-scale counting in extremely dense crowd images. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2547–2554, 2013. 8

[6] H. Idrees, M. Tayyab, K. Athrey, D. Zhang, S. Al-Maadeed, N. Rajpoot, and M. Shah. Composition loss for counting, density map estimation and localization in dense crowds. *arXiv preprint arXiv:1808.01050*, 2018. 5, 6

[7] A. Radford, L. Metz, and S. Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv preprint arXiv:1511.06434*, 2015. 3, 5

[8] T. Salimans, I. Goodfellow, W. Zaremba, V. Cheung, A. Radford, and X. Chen. Improved techniques for training gans. In *Advances in Neural Information Processing Systems*, pages 2234–2242, 2016. 3, 4

[9] D. B. Sam, S. Surya, and R. V. Babu. Switching convolutional neural network for crowd counting. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, volume 1, page 6, 2017. 5

[10] V. A. Sindagi and V. M. Patel. Cnn-based cascaded multi-task learning of high-level prior and density estimation for crowd counting. In *Advanced Video and Signal Based Surveillance (AVSS), 2017 14th IEEE International Conference on*, pages 1–6. IEEE, 2017. 5

[11] N. Souly, C. Spampinato, and M. Shah. Semi and weakly supervised semantic segmentation using generative adversarial network. *arXiv preprint arXiv:1703.09695*, 2017. 4

[12] J. T. Springenberg. Unsupervised and semi-supervised learning with categorical generative adversarial networks. *arXiv preprint arXiv:1511.06390*, 2015. 4

[13] K. Sricharan, R. Bala, M. Shreve, H. Ding, K. Saketh, and J. Sun. Semi-supervised conditional gans. *arXiv preprint arXiv:1708.05789*, 2017. 4

[14] C. Zhang, H. Li, X. Wang, and X. Yang. Cross-scene crowd counting via deep convolutional neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 833–841, 2015. 8

[15] Y. Zhang, D. Zhou, S. Chen, S. Gao, and Y. Ma. Single-image crowd counting via multi-column convolutional neural network. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 589–597, 2016. 5, 8