Crowd Counting With Minimal Data Using Generative Adversarial Networks For Multiple Target Regression

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

Abstract

In this work, we use a generative adversarial network (GAN) to train crowd counting networks using minimal data. We describe how GAN objectives can be modified to allow for the use of unlabeled data to benefit inference training in semi-supervised learning. More generally, we explain how these same methods can be used in more generic multiple regression target semi-supervised learning, with crowd counting being a demonstrative example. Given a convolutional neural network (CNN) with capabilities equivalent to the discriminator in the GAN, we provide experimental results which show that our GAN is able to outperform the CNN even when the CNN has access to significantly more labeled data. This presents the potential of training such networks to high accuracy with little data. Our primary goal is not to outpreform the state-of-the-art using an improved method on the entire dataset, but instead we work to show that through semi-supervised learning we can reduce the data required to train an inference network to a given accuracy. To this end, systematic experiments are performed with various numbers of images and cameras to show under which situations the semi-supervised GANs can improve results.

1. Introduction

A significant obstacle in many machine learning tasks is the acquisition of enough ground truth data to train a complex system. In most tasks which utilize deep neural networks, one of the most limiting factors is lack of access to enough training data. This is certainly true in the case of machine learning algorithms for crowd analysis, where labeling data is a time consuming and tedious task and where generalizing to unseen data depends on similar training data. In this work, we explore the use of generative adversarial networks in training neural networks for crowd counting with minimal amounts of data.

Generative adversarial networks (GANs) [2] have been shown to increase the accuracy of deep networks and allow for smaller quantities of data to train to high levels of accuracy [6, 7]. However, semi-supervised GANs have only been used for semi-supervised learning only in classification problems, and have not been applied to regression problems. This is both because classification problems are more commonly the focus of benchmarks in deep learning, but also because a classification problem can be simply formulated into the optimization goals of a GAN. In contrast, it becomes more difficult to assign goals for adversarial networks in regression problems, especially given the constraints of the issue in question. To the best of our knowledge an approach to using GANs for semi-supervised learning of regression targets has not previously been proposed.

In this paper, we apply GANs to the structured per pixel regression problem of generating crowd density images. We explain the challenges of adapting GANs to this regime, and provide our solutions to each challenge. More generally, our algorithm can be applied to any regression problem where the target is a continuous positive value with higher values corresponding to a stronger representation of the feature to be detected. The algorithm works both for single value regression targets, or multiple value (in our case per-pixel) regression targets. Specifically, to allow for this kind of training we design a generator which is optimized to produce the highest possible values using a novel variant of feature matching where each feature vector is weighted on the predicted value, and we propose a discriminator loss function for unlabeled and generated data allowing for training in the regression regime. Regression is a more general problem than classification. Specifically, the problem of classification is a subset of regression problems, so expanding semi-supervised GANs to regression allows for a more general understanding of their capabilities.

The primary goal of this work is not to outperform the

state-of-the-art methods when trained on a large dataset, but is to demonstrate the ability of a GAN to train a inference network to the same level of accuracy with less labeled data than is required by the inference network without the GAN. This is done by using the GAN to utilize unlabeled data as well as the small portion of labeled data. We show that this approach significantly decreases the amount of data required to train the network resulting in the same accuracy in many scenarios. Less data requirements mean less manual work in labeling and the ability to train even when large quantities of labeled data are not accessible.

To summarize, we provide the following three primary contributions: (1) We apply semi-supervised GANs to learning of the structured per pixel regression problem of generating crowd density images, and show that it can significantly reduce the amount of data required to train a predictor for crowd analysis. (2) A new formulation of GAN objectives is designed which allows GANs to deal with a set of regression problems, and showcases how regression may be approached in general. (3) A systematic study is performed in terms of the number of images and number of cameras, demonstrating the capabilities of semi-supervised GANs in various data limited circumstances.

2. Related Work

The work in [9] provides one of the first uses of convolutional neural networks (CNNs) as a method for crowd counting, especially across multiple scenes. While other works have made valuable alterations to the approach given in this paper to produce improved state-of-the-art results (such as multiple scale CNNs [8], or residual network skip connections [5]), the primary functionality of these approaches, namely the ability to use CNNs for crowd counting, is still similar to that which is presented by [9]. Our discriminator is similar to the CNN of this work, notably in the use of the joint optimization goal of two outputs of the CNN, one for count prediction and the other for density prediction. This CNN then becomes the discriminator half of our GAN. The main contributions of our work are in formulating how to allow a GAN to function properly in this set of regression target problems. We also use the datasets first presented in [9] as our experimental data.

The use of GANs for minimal data semi-supervised learning was proposed in [7]. Similar to this work, we wish to reduce the amount of data required to train the inference network. However, their approach was designed with classification problems in mind, and we need to completely redesign the optimization goals to work in the regime of problems involving multiple regression targets. The large scale goals are the same: use a GAN to allow unlabeled data to provide useful information in training a network for inference. We use the high level thinking of this approach as a starting point, but in our crowd counting case, there are no classes, and as such the loss functions described in [7] are unusable (and are unusable for regression problems in general). Our work consists of designing objectives which work for a regression target, and overcoming the obstacles in achieving such an objective.

Wasserstein GANs have been shown to produce more stable training in GANs [1, 3]. As demonstrated in these works, Wasserstein GANs theoretically converge wherever a standard GAN converges, and converge in many cases where the standard GAN does not. We found this approach to GANs essential in preventing mode collapse and avoiding unconverging states during training of our network. In our work, we train the GAN with the discriminator being trained more frequently than the generator and apply a gradient penalty to the discriminator, as is proposed in [1, 3].

In [4], a GAN is used in crowd counting. The GAN in [4] is used to improve the accuracy of the crowd counting prediction. The GAN in this work consists of a conditional generator with an input of true images and an output of generated density maps. A discriminator network then attempts to distinguish between the generated density maps and the true density maps. Once this GAN is trained, the generator is used to produce density maps of the test images, and these results are used as the predicted density maps. That is, the generator is used for inference in the testing phase. The formulation of this GAN is significantly different from our approach. Most notably, the GAN in [4] is not designed to train with unlabeled data or to specifically reduce the amount of data required. [4] expects the training images have the corresponding labels which can be used for training the discriminator. Our approach allows for the use of unlabeled data to train, with the goal of requiring significantly less labeled data. The difference in the goal of the GANs is also reflected in the structure of the networks being completely different. These differences include but are not limited to the generator outputting a density label vs an image, the generator predicting the labels vs the discriminator predicting the labels, and the generator using true images as input vs only using random noise as input.

3. Methodology

3.1. Notation

Throughout this section, we will reuse notation frequently and so we will initially define a few key elements. $\boldsymbol{x}$ refers to the input random variable. In our case, a specific value of this random variable would be an image. If the ground truth label, y, is provided, this maybe denoted as the pair $\boldsymbol{x}$, y. $\boldsymbol{x} \sim p_{data}$ denotes that $\boldsymbol{x}$ has a distribution over the real data, and $\boldsymbol{x} \sim G$ has a distribution over the fake (generated) data. The G here is the shortened notation for $G(\mathcal{N})$ where $\mathcal{N}$ is a spherical normal distribution. Specifically, this is the generator function being applied to random normal noise. In

general, G represents the generator function and D is the discriminator function. Both functions require input. For example $D(\boldsymbol{x})$ is the discriminator function applied to the input $\boldsymbol{x}$. However, as the input to the generator is always a normal distribution, we usually omit its input in the notation (i.e. G is the generator applied to the normal distribution). p_{model} denotes the probability distribution of the discriminator model. Particularly, $p_{model}(y \mid \boldsymbol{x})$ is the probability of predicting label y for the given data $\boldsymbol{x}$. Lastly, $\mathbb{E}$ denotes the expected value of a term over a probability distribution.

3.2. Semi-supervised GANs for Classification

To begin, we present how semi-supervised GANs for classification are defined. This provides a well understood basis from which we can extend the model for the regression regime.

In [7], semi-supervised learning is accomplished by having a set of examples labeled among K real classes as well as a set of examples which are unlabeled, but known to be among the K classes. A $K+1$th class which is made to represent a "fake" class is also used. In this GAN, the discriminator is optimized to give labeled examples probability distributions favoring their true class, to give unlabeled images probability distributions that favor any of the first K classes, and to give generated image distributions that favor the $K+1$th class. Conversely, the generator is optimized to have the generated images given distributions favoring the first K classes. In this way, real unlabeled images can be used to train the discriminator and allows for less labeled data. Formally, [7] describes the discriminator loss as

$$L_D = L_{supervised} + L_{unsupervised} \tag{1}$$

$$\begin{aligned} L_{supervised} = \\ -\mathbb{E}_{\boldsymbol{x},y\sim p_{data}(\boldsymbol{x},y)} \log[p_{model}(y \mid \boldsymbol{x}, y < K+1)] \end{aligned} \tag{2}$$

$$\begin{aligned} L_{unsupervised} = \\ -\mathbb{E}_{\boldsymbol{x}\sim p_{data}(\boldsymbol{x})} \log[1 - p_{model}(y = K+1 \mid \boldsymbol{x})] \\ -\mathbb{E}_{\boldsymbol{x}\sim G} \log[p_{model}(y = K+1 \mid \boldsymbol{x})]. \end{aligned} \tag{3}$$

To understand what is happening in this semi-supervised learning more intuitively, we can imagine a case of an ideal discriminator and generator. The generator would eventually learn to produce data which exactly matches the true data distribution. That is, the likelihood the generator generates any specific image is the same as the likelihood that the image exists in true data. For this to happen, the discriminator must have forced the generator to learn this, meaning the discriminator too knows exactly the data distribution. This is how using a generator may make the discriminator more accurate.

While we use the high level concepts of this approach in our own work, the objectives as outlined here are inapplicable in the case of regression targets. There can be no $N+1$th class (as there are no classes) meaning the loss functions need to be completely reformulated. Our method works to overcome this issue.

3.3. Discriminator

A label in our data is a crowd density map (an array of real numbers). Thus, instead of a cross entropy typically used in classification problems, we define our supervised loss using an $L_{1,1}$ loss (equivalent to L_1 if the matrix is flattened) over the elements of the true density labels compared with the predicted ones,

$$L_{supervised} = \mathbb{E}_{\boldsymbol{x},y\sim p_{data}(\boldsymbol{x},y)} \left\| y - D(\boldsymbol{x}) \right\|_1 . \tag{4}$$

Note that our supervised loss function omits a logarithm, as we are using a Wasserstein GAN [1]. GANs have been shown converge more consistently using a loss function based on an Earth Mover's distance (or Wasserstein distance) even among classification problems [1]. Our network also uses a discriminator training gradient penalty of Wasserstein GANs as opposed to the weight clipping proposed by [1], as weight clipping was shown to result in a GAN which produces pathological behavior while gradient penalty results in more consistent convergence over a wider range of network architectures [3].

In the classification case, there were two terms to the unsupervised loss. For clarity, in the following description of our proposed approach we define these two terms separately with

$$L_{unsupervised} = L_{unlabeled} + L_{generated}. \tag{5}$$

This way we can define the intuition for the definition of $L_{unlabeled}$ and $L_{generated}$ individually. For the case of loss from the generated images, we wish to punish the discriminator for seeing any amount of people (any density), as the generated images contain no true images of people. So here our loss is given by

$$L_{generated} = \mathbb{E}_{\boldsymbol{x}\sim G} \left\| D(\boldsymbol{x}) \right\|_1 . \tag{6}$$

With this, in a sense our equivalent of the $K+1$th class is zero density. Note however, that even real images contain areas of pixels with zero density.

As we do not know the true label for the unlabeled, training the discriminator toward an exact value can be detrimental toward the overall accuracy. For example, consider a case where we assume, based on the labeled images from the camera, that the unlabeled image has a label $\hat{y}$, but the true (unknown) label is y. If the discriminator predicts y and our loss function has no leniency, the discriminator will be trained to move away from the correct answer it predicted towards $\hat{y}$. Instead, we use a loss function which allows for a range of "correct" answers as we do not know which is true. Specifically, we use a loss function for which a range

of input values produces zero loss. That is, if the difference of the predicted from the true value is small enough, no loss is produced. However, beyond a given difference threshold, loss is non-zero. Specifically, our unlabeled loss is

$$L_{unlabeled} = \mathbb{E}_{\boldsymbol{x}\sim p_{data}(\boldsymbol{x})}\Big[\max\left(\frac{1}{\alpha}\sum y_e - \sum D(\boldsymbol{x}), 0\right) + \max\left(\sum D(\boldsymbol{x}) - \alpha\sum y_e, 0\right)\Big] \quad (7)$$

where α is an experimentally chosen hyperparameter (see Section 4) and is greater than 1. y_e is average labeling of known count labels for the labeled images being used for training (in implementation, these are the images for a given step of training). Note here that the values are summed before differences are computed. This is because there is no information on the locations of the person densities in unlabeled images and only the approximate count is useful (i.e. comparing density maps would not be).

3.4. Generator

The generator is trained on a modified version of the feature matching loss described by [7]. Once again, the main difference in our generator as compared to [7] comes from training in the case of regression targets, and more specifically, our crowd counting case. In this case, we want the generator's goal to be able to produce highly crowded examples (as interpreted by the discriminator). Here we use feature matching as the generator goal as described by [7]. Normally, feature matching is simply used to match the features that arise in the discriminator from the real images. Instead, we want features which result in the highest predictions for density and count to be the goal of the generator's matching. That is, we don't want the generator to match the features for floors or walls, but instead to match the features in areas the discriminator sees as containing crowds of individuals. If this were not the case, the discriminator and generator could simply "agree" to have the generator produce realistic, uncrowded images, as it could meet both networks' goals. To force the generator to work toward features that represent crowded portions of the images, each feature vector on the intermediate layer is weighted based on the predicted output value of that vector by the discriminator. In this way, the generator tries to produce images whose feature vectors in the discriminator match the feature vectors in the real data that have the largest count and density predictions. With $f(\boldsymbol{x})$ denoting the activations on the final layer before the output layer and $\boldsymbol{z}$ being noise to input into the generator, the generator loss is given by

$$L_G = \left\| \mathbb{E}_{\boldsymbol{x}\sim p_{data}} \left[\frac{D(\boldsymbol{x}) \odot f(\boldsymbol{x})}{\sum D(\boldsymbol{x})} \right] - \mathbb{E}_{\boldsymbol{z}\sim p_{\boldsymbol{z}}} f(G(\boldsymbol{z})) \right\|_1 \quad (8)$$

The second term of this loss function is simply getting the expected values for the activations of an intermediate layer of the discriminator for fake images. That is, the mean features of fake data. The first term does something similar for real data. The only difference is that the real features are weighted by how much crowd density they correspond to. Thus, the generator tries to produce images that have features similar to more crowded images.

3.5. Dual Optimization Goal

One additional complication comes from training a network with a second optimization goal. As shown by [9], better crowd counting results can be achieved by training a network to produce both a density map and a separate total count value. The last layer of our network is actually two layers in parallel, one for density and one for count. The density layer is trained to produce a density map which matches the true values of the label. The count layer is trained to produce values which, when summed, match the true summed value of the label. Because of this, we actually have two losses for both discriminator and generator. Luckily, other than which output of the discriminator is used, all the losses are identical in both cases except the labeled loss of the discriminator for the case of the count. That loss is given by

$$L_{supervised} = \mathbb{E}_{\boldsymbol{x},y\sim p_{data}(\boldsymbol{x},y)} \left[\sum y - \sum D(\boldsymbol{x}) \right]. \quad (9)$$

Note that the only difference is that rather than the norm distance between the arrays, the loss is the difference of the sum of the arrays. We use the same network for our CNN as [9].

4. Implementation

In this work we use the Shanghai Jiao Tong University WorldExpo'10 crowd counting dataset [9]. The process of selecting cameras for use in training is done entirely randomly, and this random selection process was not repeated based on trial results. The number of cameras and images used for each experiment is given in Section 5.

The original dataset consists of images, head positions, a region of interest within which head positions were labeled, and average person height at any given point within the image. Using this, we created a crowd density map for each image. This was done by creating two 2D gaussian distributions which approximately fit the shape of a human, with one gaussian covering the head and the other covering the body. This gaussian was sized according the average person height for each head position. The sum of the values of the two gaussians is equal to 1. This way, the sum of the entire density map label is equivalent to the number of individuals within that image. An example of the images in the dataset as well as the corresponding labels can be seen in Figure 1.

Figure 1. Examples of the data images and corresponding density labels.

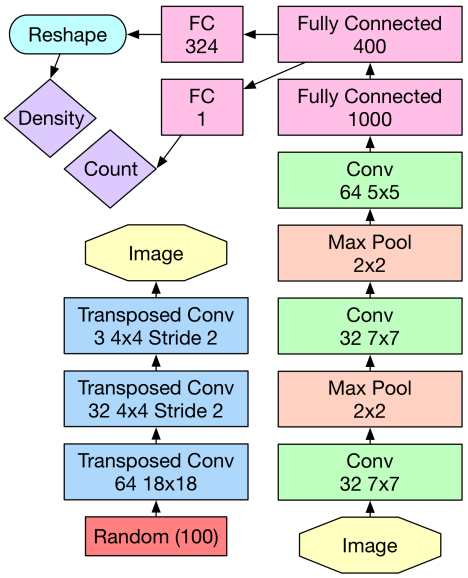

Figure 3. The structure of both the generator (left) and the discriminator (right).

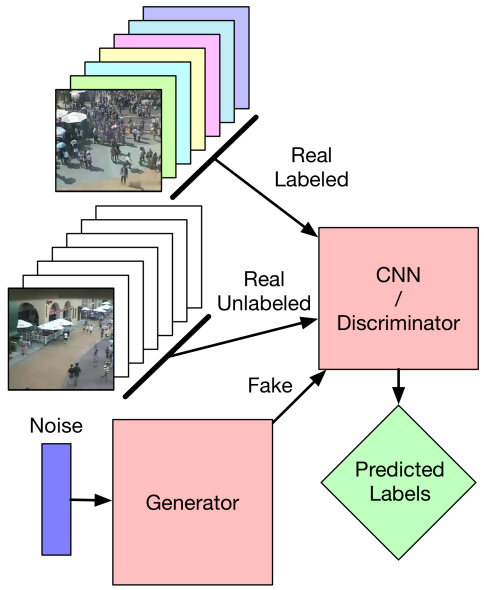

Figure 2. The structure of the GAN as a whole. In the case where the GAN is not being used, only the CNN and labeled data is present.

The overall structure of the generative adversarial network as a whole can be seen in Figure 2.

During these experiments, we used a generator with a normal noise input vector containing 100 values. This is passed through 3 transposed convolutional layers with output channels 64, 32, and 32 and kernel size 18, 4, and 4. Additionally a stride of 2 is used on the second and third transposed convolutional layers. Note, first transpose convolutional layer happens on the noise vector, which has only 1 spatial dimension, and is then equivalent to a fully connected layer with the output layer being reshaped to 64x18x18. All layers use a leaky ReLU activation except the final layer which uses a TanH activation to push the values into the -1 to 1 range which is what the image pixel values are normalized to.

The discriminator is equivalent to the CNN used by [9]. First there are 3 convolutional layers with output channels 32, 32, and 64 and with kernel size 7, 7, and 5. Between the convolutional layers are max pooling layers with kernel size 2. After, there are 2 fully connected layers with output channels 1000 and 400. The final two layers happen in parallel working on the 400 channel output layer. One has 324 output channels and the other has a single output channel. The first corresponds to a density map and is reshaped to an 18x18 patch. The other is a value which estimates the total count output in the entire image patch. Each layer uses a leaky ReLU as the activation function.

The generator and discriminator networks are visualized in Figure 3.

The original image dimensions are 720x576. Each image used in training is a randomly selected patch of the original image. A patch is approximately 3x3 meters in the real world. This size is determined by the perspective map provided in the original database. Then this image patch is resized to 72x72 pixels when input into the network. During testing, patches from each full test image are taken with approximately 50% overlap between patches. The resulting crowd densities are averaged where patches overlap to produce a final predicted count.

An Adam optimizer is used for training. The images were augmented by random horizontal flipping. In the fash-

ion of the Wasserstein GAN, we allow the discriminator to train to convergence between steps of training the generator (the value of this demonstrated and explained [1]). We additionally apply a gradient penalty to the discriminator during GAN training (the value of this is demonstrated and explained in [3]). The α defined in the unlabeled loss from methodology was set to 2 during our experiments.

When training the GAN, each step of training uses the labeled, unlabeled, and fake (generated) images. For each, a equal batch size of images is used on each training step for each type of data (i.e. N labeled, N unlabeled, and N fake). Both the labeled and unlabeled datasets are finite. When all the images of one of these two datasets have been cycled through, it repeats, though with a new shuffling of the data. The fake images are newly generated on each step, as the generator is changing during training.

All code, including data preprocessing and hyperparameters, can be found at *(repository link will be provided upon paper acceptance)*.

5. Experiments

5.1. Evaluation Protocol

The Shanghai Jiao Tong University WorldExpo'10 crowd counting dataset is used for our experiments. The dataset includes images and videos from 109 cameras, and consists of a training dataset and a testing dataset. We used the WorldExpo'10 training data, and from it generated a set of training datasets, and a single validation dataset, and a single testing dataset (in addition to the original WorldExpo'10 testing dataset). Our training datasets (as a group), validation dataset, and each testing dataset are entirely disjoint from one another, in particular disjoint in their use of cameras/scenes. CNN and GAN models are trained using a varying number of cameras (1, 3, 5, 10 and 20) and varying number of images (1, 3, 5, 10 and 20) per camera in order to systematically compare the performance of the two models; details of the selections will be provided in Section 5.2. However, the same testing datasets and validation dataset are used for all the trained models.

One testing dataset is simply the WorldExpo'10 testing dataset, and it consists of 600 images from 5 cameras. We want to note that no images from cameras in the testing dataset are included in the training dataset and vice versa. That is, training and testing is preformed across scenes. The datasets are disjoint in camera source selection. Each camera from this test dataset has 120 images. This dataset contains 2 of the most crowded scenes (in the entire WorldExpo'10 database), 2 of the least crowded scenes (and 1 additional scene). While this provides a good challenging case to test on, it is not very representative of the training data. As such, we prepared the additional testing dataset. This provides better insight into how accurate the methods will be when testing on data whose statistics more closely match the training data statistics (i.e. this dataset is specifically chosen randomly from the same data generating distribution, rather than hand chosen to be a challenging case). The second testing dataset is from a subset of the WorldExpo'10 training dataset. This testing dataset consists of 370 images from 10 cameras. These cameras were randomly chosen from among the full dataset before any models were trained. No images from cameras in the validation dataset are included in the training dataset and vice versa. Again, the two datasets are disjoint camera source selection as well.We note that in general results are better on our randomly chosen testing dataset then on the original testing dataset. Finally, the validation dataset consists of a different 307 images from 10 cameras, and was only used to try different models and examine how well they generalize without comparing against the final testing datasets.

In each of the following experiments, we train both a CNN model and a GAN model on the same set of labeled data (for each of the camera-image number combinations). The GAN additionally uses the unlabeled images from the cameras in the trial's training dataset (the CNN has no way to profitably use this data). Each resulting model is used to predict the person count of each image in both test datasets. The accuracies between the two methods are compared. In every case, the GAN uses as its discriminator a network identical to the CNN. As we compare the two methods using a varying number of cameras and varying number of images per camera, this gives both an understanding of the amount of total image data required by each network as well as the distinct scene information (number of cameras) required by each network.

The unlabeled data from the GAN model consists of video from which the labeled image data was taken for training. The GAN allows for any number of these frames to be used with no additional labeling (which is the primary advantage of the GAN). As such, in these experiments, we used all (unlabeled) video data for any camera within the training set (that is, the number of cameras the GAN has access to is still limited). Though the video lengths vary, on average there is approximately 2 minutes of video for each camera with 50 FPS. It should be noted that as this is video data, many of the frames are near identical to others.

5.2. Semi-Supervised Results

We demonstrate the ability of a GAN to train an inference network using less labeled data than would be required to achieve the same level of accuracy by an identical inference network outside a GAN. The process of selecting cameras is done entirely randomly, and this random selection process was not repeated based on trial results. For each set of experiments, the cameras and images chosen are consistent between experiments. That is, when 10 cameras are used,

the first 5 of these cameras are the ones used in the 5 camera trial. The same is true for the images used. Specifically, in the table, the training dataset used for any trial is a subset of the training dataset for the trial to the right and below it on the table. Although cameras were chosen randomly, to allow for reproducibility, the list of cameras and images used is provided in the appendix (supplementary material). In the GAN cases, the unlabeled image data which is used is only from the cameras which are in the labeled set for that trial (i.e. no cameras are included which are not included in that training set). In all the training cases listed, the networks are trained for 7000 epochs.

We tested the trained models using a varying number of cameras with a varying number of images. For each case, training using the CNN (discriminator) alone and training with the GAN is compared. These results can be seen in Tables 1 and 2. For each experiment, the number of cameras is given along with the number of images used per camera. The data is explicitly limited so that we can experiment on how much data is needed to train the system to different levels of accuracy, and to compare how much data is needed with the generator to how much is needed without it. The numbers of images and cameras used are referring to the training dataset only. In all trials the entirety of both testing datasets are used. Table 1 shows the results on the randomly chosen test dataset and Table 2 are the results on the original WorldExpo'10 test dataset. Again, there is no overlap of cameras between the test datasets and the training datasets (or between the two test datasets).

When trained using the entire database of available data, the CNN and GAN reach an accuracy of 11.1 and 14.2 for the random test dataset, and 19.7 and 23.9 for the original test dataset. Training using the entire dataset, the CNN performs better. The CNN outperforming the GAN in this case is likely due to the GAN introducing a bias and the labeled data samples for training CNN is already sufficient.

5.3. Analysis

We have made the following interesting observations.

(1) When data is limited, training using the semi-supervised GAN usually allows the network to train to higher accuracies than the CNN with the same, limited amount of labeled data. This is true on both the random test dataset and the original test dataset, even though the errors on the random test dataset are much lower than on the testing dataset (again, due to the random test dataset more closely matching the training dataset distribution).

(2) The effectiveness of the GAN is most pronounced when the labeled data is most limited. This makes sense as what the semi-supervised GAN does is to provide a form of regularization that is based on real, but unlabeled data. It discourages the discriminator from overfitting the data as it's method needs to have reasonable results on the much larger unlabeled dataset as well. This suggests that much more powerful networks could be used while mitigating the likelihood of overfitting. When we reach approximately 50-60 total images, the CNN begins to occasionally outperform the GAN. As noted, when training using the entire dataset, the CNN performs better. The CNN outperforming the GAN in these cases is likely due to the GAN introducing a bias. Specifically, the expected value of the unlabeled data is greater than zero. As this introduces a bias in the training, taken to the extreme of unlimited data, the bias will create error. The stronger the bias introduced, the lower amounts of data that the unbiased network will begin to outperform the biased network. Though speculative, we believe a loss function which removes this bias would remove the advantage of the CNN while keeping the advantage of the GAN. This might be done using feature matching of a feature layer instead of count matching on the unlabeled data, but this is left for future work.

(3) In general, the increase in number of cameras reduces the error in both CNN and GAN cases. With the same number of total training images, the error tends to be lower with more cameras. This is due to additional scenes, lighting conditions, etc being included in these cases to be generalized to the testing cameras which are not seen in training. Additional images usually only give additional configurations of individuals in the images. This also suggests that using unlabeled data from unseen cameras in the GAN may provide the GAN a significant increase in advantage, but this is left for future work.

(4) The estimation errors on both testing datasets mostly exhibit consistent trends with regards to the numbers of cameras and images, and for both CNN and GAN. However we have noticed a few outliers on both the testing datasets: For example, in both testing data, the errors of both GAN and CNN trained on 5 cameras x 5 images is significantly higher than would be expected given trend of surrounding data points. In both datasets, the GAN does very poorly with only a single image from a single camera. At the same time, the CNN does poorly with several images from a single camera. Particularly when dealing with only a single camera, the exact images in question and luck during training may play a significant role.

6. Conclusions

In this work, we've used GANs to train crowd counting with less data than is required to train an equivalent inference network. We've demonstrated a case of how to allow GAN based semi-supervised learning to be usable in a multiple target regression problem. We've used a presented a weighted feature matching approach and provided a lenient unlabeled loss goal. We have given experimental results to show that a GAN network outperforms an equivalent CNN using significantly less data. Our ongoing work include the use of

Random Test Dataset Error						
Number of training cameras		Number of training images per camera				
		1	3	5	10	20
1	CNN	93.2	116.8	131.5	25.2	32.0
	GAN	134.0	70.1	37.5	32.4	19.9
3	CNN	46.8	29.1	17.4	25.0	21.7
	GAN	26.1	28.0	28.8	22.4	19.9
5	CNN	31.3	29.2	48.9	13.5	
	GAN	22.2	24.4	27.9	18.2	
10	CNN	31.0	17.8	17.4		
	GAN	29.4	24.3	17.5		
20	CNN	26.1	25.8			
	GAN	22.7	24.1			

Table 1. A table of the mean absolute error when the network is trained with varying amounts of data with or without the generator. For each experiment, the number of cameras is given along with the number of images used per camera. Additionally, it is shown whether the GAN or the plain CNN is used. The test dataset is the same for every case

Original WorldExpo'10 Test Dataset Errorr						
Number of training cameras		Number of training images per camera				
		1	3	5	10	20
1	CNN	65.8	168.5	189.2	32.8	45.9
	GAN	169.5	74.3	53.5	36.0	27.9
3	CNN	73.3	60.3	35.0	29.6	30.2
	GAN	112.6	36.4	30.5	32.3	32.9
5	CNN	58.0	55.5	68.1	23.7	
	GAN	33.6	40.1	42.8	30.2	
10	CNN	52.3	25.7	24.6		
	GAN	40.0	31.5	26.8		
20	CNN	46.6	38.7			
	GAN	39.5	32.7			

Table 2. A table of the mean absolute error when the network is trained with varying amounts of data with or without the generator. For each experiment, the number of cameras is given along with the number of images used per camera. Additionally, it is shown whether the GAN or the plain CNN is used. The test dataset is the same for every case

loss functions which do not include a bias (which leads to the CNN outperforming the GAN on the full dataset), and compare the performance of our GAN model with the state-of-the art methods.

7. Acknowledgments

This research was performed under appointments to the U.S. Department of Homeland Security (DHS) Science & Technology Directorate Office of University Programs, administered by the Oak Ridge Institute for Science and Education (ORISE) through an interagency agreement between the U.S. Department of Energy (DOE) and DHS. ORISE is managed by ORAU under DOE contract number DE-AC05-06OR23100 and DE-SC0014664. All opinions expressed in this paper are the author's and do not necessarily reflect the policies and views of DHS, DOE, or ORAU/ORISE. This work is also supported by the U.S. Computing resources were supported by Azure for Research. National Science Foundation through Award EFRI-1137172 and SCC-Planning-1737533. Additional support by a CUNY-Bentley CRA.

References

[1] Martin Arjovsky, Soumith Chintala, and Léon Bottou. "Wasserstein gan". In: *arXiv preprint arXiv:1701.07875* (2017).

[2] Ian Goodfellow et al. "Generative adversarial nets". In: *Advances in neural information processing systems*. 2014, pp. 2672–2680.

[3] Ishaan Gulrajani et al. "Improved training of wasserstein gans". In: *arXiv preprint arXiv:1704.00028* (2017).

[4] Jiawen Li et al. "An end-to-end generative adversarial network for crowd counting under complicated scenes". In: *Broadband Multimedia Systems and Broadcasting (BMSB), 2017 IEEE International Symposium on*. IEEE. 2017, pp. 1–4.

[5] Mark Marsden et al. "ResnetCrowd: A Residual Deep Learning Architecture for Crowd Counting, Violent Behaviour Detection and Crowd Density Level Classification". In: *arXiv preprint arXiv:1705.10698* (2017).

[6] Alec Radford, Luke Metz, and Soumith Chintala. "Unsupervised representation learning with deep convolutional generative adversarial networks". In: *arXiv preprint arXiv:1511.06434* (2015).

[7] Tim Salimans et al. "Improved techniques for training gans". In: *Advances in Neural Information Processing Systems*. 2016, pp. 2234–2242.

[8] Lingke Zeng et al. "Multi-scale Convolutional Neural Networks for Crowd Counting". In: *arXiv preprint arXiv:1702.02359* (2017).

[9] Cong Zhang et al. "Cross-scene crowd counting via deep convolutional neural networks". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2015, pp. 833–841.